

به نام خدا



دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه درس بازیابی اطلاعات

(OpenSearch)

استاد درس:

دکتر محمد مهدی رضاپور

ردیف	اعضای گروه :	شماره دانشجویی:	مشارکت در انجام پروژه:
۱	مهرو سادات نوحی	۹۹۳۶۱۳۰۶۱	توضیحات عملی انواع سرچ و روش های جستجو و چگونگی پیاده سازی و کدنویسی آنها
۲	زهرا امیری نژاد	۹۹۳۶۱۳۰۰۸	کد نویسی فرانت پروژه و توضیحات مربوط به امنیت تا پایان تجمعات طبق فهرست مطالب
۳	یاسمین اکبری	۹۹۳۶۱۳۰۰۶	توضیحات مربوط به قسمت های تاریخچه تا امنیت طبق فهرست مطالب و نگارش و تهیه داکيومنت نهایی پروژه

نیم سال دوم تحصیلی ۰۳-۰۲

فهرست:

۵	تاریخچه و پیدایش:
۶	Elasticsearch vs. OpenSearch:
۶	Elasticsearch چیست؟
۶	تنش های بین Elastic و AWS:
۷	تفاوت های کلیدی بین Elasticsearch و OpenSearch:
۹	Elasticsearch vs. OpenSearch چگونه انتخاب کنیم؟
۹	مفاهیم کلیدی OpenSearch:
۹	موتور جستجو: OpenSearch چگونه جستجو را انجام می دهد؟
۱۰	تجزیه و تحلیل داده: OpenSearch چه نوع تجزیه و تحلیل داده ای را ارائه می دهد؟
۱۰	مقیاس پذیری: OpenSearch چگونه مقیاس بندی می شود؟
۱۱	قابلیت استفاده: OpenSearch چقدر کاربر پسند است؟
۱۱	متن باز: OpenSearch چه معنای متن باز بودن دارد؟
۱۲	قابلیت های کلیدی:
۱۵	مهاجرت از Elasticsearch OSS به OpenSearch: (اطلاعات بیشتر)
۱۵	مدیریت ایندکس در OpenSearch: (اطلاعات بیشتر)
۱۶	ملاحظات کلیدی در هنگام ایندکس کردن اسناد:
۱۷	معایب تولید خودکار شناسه (ID)
۱۸	Read data:
۱۸	بروزرسانی اطلاعات:
۱۹	حذف داده ها:
۲۰	قالب های ایندکس:
۲۰	نام مستعار شاخص (Index aliases):
۲۰	جریان های داده (Data Streams):
۲۱	مدیریت وضعیت ایندکس (Index State Management):
۲۱	تبدیل ایندکس (Transform Jobs):
۲۲	خلاصه سازی ایندکس (Index Rollups)
۲۳	امنیت مدیریت ایندکس
۲۴	OpenSearch Dashboards: (اطلاعات بیشتر)
۲۴	قابلیت های کلیدی OpenSearch Dashboards:
۲۴	مزایای استفاده از OpenSearch Dashboards:
۲۵	OpenSearch Dashboards برای چه کسانی مناسب است؟
۲۵	شروع کار با OpenSearch Dashboards:

۲۵	امنیت در OpenSearch (اطلاعات بیشتر).....
۲۵	ویژگی ها در یک نگاه
۲۶	رمزگذاری
۲۶	احراز هویت
۲۶	کنترل دسترسی
۲۷	ثبت حسابرسی و انطباق
۲۷	سایر ویژگی ها و قابلیت ها
۲۷	داشبوردهای چند اجاره ای
۲۸	جستجوی متقابل خوشه ای
۲۸	تجزیه و تحلیل امنیتی
۲۸	منابع و اطلاعات
۲۸	اجزا و مفاهیم
۲۸	آشکارسازها
۲۸	انواع لاگ
۲۹	قوانین تشخیص
۲۹	یافته ها
۲۹	هشدارها
۲۹	موتور همبستگی
۳۰	نگاشت ها و انواع فیلدها
۳۱	نقشه برداری پویا
۳۲	نقشه برداری صریح
۳۳	تحلیل متن
۳۳	آنالایزرها
۳۴	آنالایزهای داخلی
۳۵	تست یک آنالایزر
۳۶	پرس و جو از DSL
۳۶	پرس و جوهای برگ
۳۷	تجمعات (Aggregations)
۳۷	تجمع در فیلدهای متنی
۳۸	ساختار تجمع عمومی
۴۰	داکیومنت:
۴۰	اینکدس:
۴۱	جدول معکوس:
۴۳	بخش عملی ارتباط با open search

۴۵	زبان‌های مورد پشتیبانی برای اتصال Open search
۴۸	ایجاد شاخص
۴۹	متد save():
۵۰	متد bulk:
۵۰	متد client.delete():
۵۱	رابط کاربری:
۵۲	بازیابی اطلاعات:
۵۲	جستجو کردن:
۵۲	روش های جستجو
۵۳	فراگیری ماشین (Machine Learning)
۵۳	یکپارچه سازی مدل های ML:
۵۳	مدیریت مدل های ML در داشبوردهای OpenSearch:
۵۳	پشتیبانی از الگوریتم ها:
۵۳	ML Commons API:
۵۴	زبان‌های جستجوی در OpenSearch:
۵۴	عملکرد جستجو
۵۵	Searching data
۵۵	1. Paginate results
۵۷	2. Sort results
۵۸	3. Autocomplete functionality
۶۲	4. Did-you-mean
۶۴	Keyword search
۶۶	جستجوی مبتنی بر یادگیری ماشینی (ML)
۶۶	k-NN search
۷۶	جستجوی عصبی:
۷۹	سرچ عصبی به صورت عملی:

تاریخچه و پیدایش:

پروژه OpenSearch در آوریل ۲۰۲۱ توسط آمازون وب سرویسز (AWS) معرفی شد. این پروژه به عنوان یک واکنش به تغییرات در سیاست‌های لایسنس Elasticsearch و Kibana توسط شرکت Elastic NV ایجاد شد. تا پیش از این تغییرات، Elasticsearch و Kibana با لایسنس Apache 2.0 عرضه می‌شدند که یک لایسنس متن‌باز شناخته‌شده و محبوب است. با این حال، در اوایل سال ۲۰۲۱، شرکت Elastic NV تصمیم گرفت تا لایسنس این پروژه‌ها را به SSPL (Server Side Public License) تغییر دهد. این تغییرات باعث نگرانی‌هایی در جامعه متن‌باز و همچنین در میان کاربران سازمانی شد که از Elasticsearch و Kibana استفاده می‌کردند.

AWS به عنوان یکی از بزرگ‌ترین ارائه‌دهندگان خدمات ابری و یکی از بزرگ‌ترین کاربران Elasticsearch، به سرعت به این تغییرات واکنش نشان داد. AWS تصمیم گرفت که یک فورک از نسخه ۷.۱۰.۲ Elasticsearch و Kibana ایجاد کند و این پروژه‌ها را تحت نام OpenSearch و OpenSearch Dashboards به صورت متن‌باز و با لایسنس Apache 2.0 عرضه کند. هدف اصلی AWS از این اقدام، حفظ و تداوم جامعه متن‌باز و ارائه یک جایگزین پایدار و قابل اعتماد برای Elasticsearch و Kibana بود که همچنان به صورت کاملاً متن‌باز در دسترس باشد.

با معرفی OpenSearch، AWS متعهد شد که به توسعه و بهبود این پروژه به همراه جامعه متن‌باز ادامه دهد. این تعهد شامل افزودن ویژگی‌های جدید، بهبود کارایی، و ارائه به‌روزرسانی‌های امنیتی بود. همچنین، AWS اعلام کرد که تمامی کدها و تغییرات در OpenSearch به صورت عمومی در GitHub در دسترس خواهد بود و از مشارکت جامعه متن‌باز استقبال خواهد شد. این اقدام باعث شد که OpenSearch به سرعت جای خود را در میان کاربران قبلی Elasticsearch و Kibana پیدا کند و به عنوان یک جایگزین معتبر و مورد اعتماد شناخته شود.

یکی از دلایل موفقیت سریع OpenSearch، پشتیبانی قوی AWS از این پروژه بود. AWS نه تنها منابع مالی و فنی لازم را برای توسعه و نگهداری OpenSearch فراهم کرد، بلکه از توانمندی‌های زیرساختی خود نیز برای ارائه این سرویس در قالب Amazon OpenSearch Service بهره برد. این سرویس، که پیش‌تر با نام Amazon Elasticsearch Service شناخته می‌شد، به کاربران امکان می‌دهد تا به راحتی OpenSearch را در محیط‌های ابری AWS اجرا و مدیریت کنند. این یکپارچگی با سرویس‌های AWS، یکی از عوامل مهمی بود که باعث شد OpenSearch به سرعت مورد استقبال قرار گیرد.

در ادامه توسعه **OpenSearch** ، **AWS** و جامعه متن باز بر روی افزودن ویژگی‌های جدید و بهبود قابلیت‌های موجود تمرکز کردند. از جمله این ویژگی‌ها می‌توان به بهبودهای امنیتی، افزودن قابلیت‌های پیشرفته جستجو و تحلیل داده‌ها، و بهبود کارایی و مقیاس‌پذیری اشاره کرد. همچنین، پلاگین‌ها و اکستنشن‌های متنوعی برای **OpenSearch** توسعه داده شد که امکان گسترش و سفارشی‌سازی این سیستم را برای کاربران فراهم می‌کند.

به طور خلاصه، **OpenSearch** به عنوان یک پروژه متن باز و با پشتیبانی قوی **AWS**، توانست به سرعت جایگاه خود را در میان کاربران پیدا کند و به یک ابزار قدرتمند و مورد اعتماد برای جستجو و تحلیل داده‌ها تبدیل شود. این پروژه با تمرکز بر ارائه یک سیستم پایدار و قابل اعتماد، همچنان به توسعه و بهبود خود ادامه می‌دهد و جامعه متن باز نیز نقش مهمی در این فرایند ایفا می‌کند.

Elasticsearch vs. OpenSearch:

Elasticsearch چیست؟

Elasticsearch یک موتور جستجوی توزیع‌شده، متن باز و مبتنی بر **REST** است که بر اساس **Apache Lucene** ساخته شده است. این موتور برای مدیریت حجم بالای داده‌ها طراحی شده و به همین دلیل انتخاب محبوبی برای مدیریت داده‌های لاگ و رویدادها است. **Elasticsearch** به خاطر قابلیت‌های بلادرنگ خود شناخته شده است و به کاربران این امکان را می‌دهد که الگوهای داده‌ها را به محض وقوع بررسی، تحلیل و مصور کنند.

علاوه بر مدیریت داده‌های لاگ و رویدادها، **Elasticsearch** معمولاً برای جستجوی متن کامل و کاربردهای هوش عملیاتی استفاده می‌شود. هدف آن این است که مقیاس‌پذیر، مقاوم و سریع باشد، و امکان فهرست‌بندی و جستجوی داده‌ها در زمان تقریباً واقعی را فراهم کند. همچنین از پرس‌وجوهای پیچیده برای انجام تحلیل‌های دقیق و پشتیبانی از چندگانگی (**multi-tenancy**) برای مدیریت آسان چندین شاخص (**indices**) پشتیبانی می‌کند.

یکی از ویژگی‌های کلیدی **Elasticsearch** طبیعت توزیع‌شده آن است. این بدان معنی است که شاخص‌ها می‌توانند به چندین شار (**shard**) تقسیم شوند که هر شار به صورت یک شاخص مستقل عمل می‌کند. این ویژگی به تسهیل مدیریت مجموعه‌های بزرگ داده‌ها با توزیع آن‌ها در یک خوشه از سرورها کمک می‌کند.

تنش‌های بین Elastic و AWS :

Elasticsearch که در سال ۲۰۱۰ تحت مجوز **Apache 2.0** منتشر شد، به موتور جستجوی ارجح در سطح جهانی تبدیل شد. معمولاً با **Logstash** و **Kibana** که با هم به نام **ELK stack** شناخته می‌شوند (استفاده می‌شود و در کاربردهای تحلیل لاگ مانند مانیتورینگ اپلیکیشن‌ها، تحلیل لاگ‌های امنیتی و ردیابی رفتار کاربران عالی عمل می‌کند). **Amazon** با شناخت این پتانسیل، سرویس **Amazon Elasticsearch Service (Amazon ES)** را در سال ۲۰۱۵

معرفی کرد که یک سرویس ابری مدیریت شده بود و به کاربران AWS اجازه می داد خوشه های مقیاس پذیر Elasticsearch را مستقر و فعالیت های داده ای خود را در ابر مدیریت کنند.

با این حال، تنش ها بین Elastic N.V.، شرکت پشت Elasticsearch، و Amazon افزایش یافت. Elastic N.V.، Amazon را به نقض علائم تجاری و بازاریابی همراه کننده متهم کرد و در سال ۲۰۱۹ شکایتی مطرح کرد. این اختلاف در سال ۲۰۲۱ به تحولات مهمی منجر شد: در ژانویه، Elastic N.V. مجوز Elasticsearch را به Server Side Public License (SSPL) و مجوز Elastic با نسخه ۷.۱۱ تغییر داد. این اقدام به منظور جلوگیری از ارائه Elasticsearch به عنوان یک سرویس بدون همکاری با Elastic توسط شرکت ها از جمله Amazon بود.

در پاسخ، Amazon آخرین نسخه متن باز (7.10.2) Elasticsearch را در آوریل ۲۰۲۱ شاخه کرد و پروژه متن باز جدیدی به نام OpenSearch را آغاز کرد. در کنار این، Amazon سرویس Amazon OpenSearch را معرفی کرد، بنابراین از تغییرات مجوز عبور کرد و به ارائه یک راه حل موتور جستجو به مشتریان ابری خود ادامه داد.

تفاوت های کلیدی بین Elasticsearch و OpenSearch :

در حالی که Elasticsearch و AWS OpenSearch از یک نسل مشترک و عملکرد اصلی برخوردار هستند، تفاوت های کلیدی بین آن ها وجود دارد که آن ها را متمایز می کند.

۱. **ورود داده ها :** هر دو Elasticsearch و AWS OpenSearch قابلیت های قدرتمند ورود داده ها را ارائه می دهند، اما به روش های متفاوتی به این کار می پردازند. Elasticsearch از انواع و ساختارهای مختلف داده پشتیبانی می کند و از نودهای ورود برای پردازش اولیه مستندات قبل از فهرست بندی استفاده می کند. همچنین از ورود داده به صورت عمده پشتیبانی می کند که آن را به انتخاب خوبی برای تحلیل داده های بزرگ مقیاس تبدیل می کند. OpenSearch بر سهولت استفاده و یکپارچگی با دیگر سرویس های AWS تمرکز دارد. این سرویس یک پایپلاین مدیریت شده برای ورود داده ها ارائه می دهد که فرآیند انتقال داده به سیستم را ساده می کند. AWS OpenSearch همچنین با سرویس هایی مانند AWS Kinesis، AWS Glue و AWS Lambda به خوبی یکپارچه می شود و یک پایپلاین پردازش داده کامل ارائه می دهد.

۲. **کتابخانه های کلاینت:** Elasticsearch دارای طیف وسیعی از کتابخانه های کلاینت در بسیاری از زبان های برنامه نویسی مانند Java، Python، .NET، PHP، Perl، و Ruby است. این امر به توسعه دهندگان اجازه می دهد تا Elasticsearch را به راحتی در برنامه های خود ادغام کنند، صرف نظر از زبان برنامه نویسی که استفاده می کنند.

AWS OpenSearch نیز مجموعه ای از کتابخانه های کلاینت خود را دارد. تا زمان نگارش این مطلب، کلاینت هایی برای Python، Java، JavaScript (Node.js)، Go، Ruby، PHP، .NET، و Rust ارائه

می‌دهد. علاوه بر این، OpenSearch به صورت فنی با کلاینت‌های Elasticsearch سازگار است زیرا اساساً همان پلتفرم Elasticsearch است. با این حال، Elasticsearch محدودیت‌های مجوزی اضافه کرده است که کلاینت‌های آن را از اتصال به OpenSearch منع می‌کند.

۳. **عملکرد:** هر دو Elasticsearch و AWS OpenSearch برای مدیریت حجم زیادی از داده‌ها و ارائه نتایج جستجوی سریع و قابل اعتماد طراحی شده‌اند. آن‌ها از موتور زیربنایی مشابه (Lucene) استفاده می‌کنند و ویژگی‌های مشابهی مانند شارده‌سازی، تکرار و معماری توزیع‌شده برای اطمینان از عملکرد بالا را ارائه می‌دهند. با این حال، OpenSearch به دلیل اینکه بخشی از یک سرویس مدیریت‌شده کامل است، از زیرساخت جهانی AWS برای بهبود عملکرد، مقیاس‌پذیری و قابلیت اطمینان بهره می‌برد AWS. ابزارهای نظارت بر عملکرد، پشتیبان‌گیری خودکار و ویژگی‌های بازایی پس از فاجعه را به عنوان بخشی از سرویس OpenSearch ارائه می‌دهد که به اطمینان از عملکرد بالا و ایمنی داده‌ها کمک می‌کند.

۴. **مجوزدهی و قیمت‌گذاری:** در زمینه مجوزدهی، هر دو Elasticsearch و OpenSearch در سال‌های اخیر تغییرات قابل توجهی را تجربه کرده‌اند. در سال ۲۰۲۱، Elasticsearch مجوز خود را از Apache 2.0 به Server Side Public License (SSPL) تغییر داد. این تغییر باعث ایجاد بحث و جدل در جامعه متن‌باز شد زیرا SSPL به عنوان یک مجوز متن‌باز توسط Open Source Initiative (OSI) شناخته نمی‌شود. در پاسخ، AWS آخرین نسخه مجوز Apache از Elasticsearch را شاخه کرد تا OpenSearch را ایجاد کند که تحت مجوز Apache 2.0 باقی می‌ماند.

مدل‌های قیمت‌گذاری Elasticsearch و OpenSearch نیز تفاوت‌هایی دارند Elasticsearch.، که توسط Elastic مدیریت می‌شود، مدل قیمت‌گذاری چندلایه‌ای ارائه می‌دهد. این شامل یک لایه رایگان با ویژگی‌های پایه و لایه‌های پولی است که قابلیت‌های پیشرفته‌تری را باز می‌کنند OpenSearch.، به عنوان یک پروژه مدیریت‌شده توسط AWS، در تمام سطوح کارکرد رایگان است. با این حال، کاربران در صورت استفاده از سرویس‌های AWS برای میزبانی و مدیریت موارد OpenSearch خود هزینه‌هایی را متحمل می‌شوند.

۵. **پشتیبانی و مستندات:** Elasticsearch مجموعه‌ای غنی از مستندات را ارائه می‌دهد که همه چیز از تنظیمات اولیه تا سناریوهای پیشرفته استفاده را پوشش می‌دهد. همچنین جامعه بزرگی و فعالی دارد که می‌تواند پشتیبانی ارائه دهد Elastic.، شرکت پشت Elasticsearch، گزینه‌های پشتیبانی پولی نیز ارائه می‌دهد.

OpenSearch یک پروژه نسبتاً جدید است و مستندات آن هنوز در حال رشد است. با این حال، AWS تعهد دارد که مستندات جامع برای OpenSearch را نگه دارد و مستندات موجود Elasticsearch تا حد زیادی قابل استفاده باقی می‌ماند OpenSearch. همچنین از پشتیبانی جامعه گسترده AWS بهره‌مند است. مشابه Elastic، AWS گزینه‌های پشتیبانی پولی برای OpenSearch ارائه می‌دهد.

۶. امنیت: Elasticsearch در ابتدا ویژگی‌های امنیتی پیشرفته را تنها در لایه‌های پولی خود ارائه می‌داد. با این حال، پس از تغییر مجوز، Elastic اعلام کرد که این ویژگی‌ها رایگان خواهند بود. این ویژگی‌ها شامل رمزگذاری SSL، کنترل دسترسی مبتنی بر نقش و ثبت گزارش‌های نظارتی است.

OpenSearch، در مقابل، ویژگی‌های امنیتی را به عنوان بخشی از بسته خود ارائه می‌دهد، به شرطی که آن را روی AWS اجرا کنید. این ویژگی‌ها شامل رمزگذاری، احراز هویت کاربران و کنترل دسترسی دقیق است. با توجه به مدیریت آن توسط AWS، کاربران می‌توانند از زیرساخت امنیتی و مطابقت قدرتمند AWS cloud نیز بهره‌مند شوند.

Elasticsearch vs. OpenSearch چگونه انتخاب کنیم؟

انتخاب بین Elasticsearch و OpenSearch تا حد زیادی به نیازها و شرایط خاص شما بستگی دارد. اگر محصولی بالغ با جامعه‌ای وسیع و مستندات گسترده را ترجیح می‌دهید، Elasticsearch ممکن است گزینه بهتری باشد. از سوی دیگر، اگر به اصول نرم‌افزار متن‌باز ارزش می‌دهید و ابزاری را ترجیح می‌دهید که ویژگی‌های پیشرفته را به صورت رایگان ارائه می‌دهد، OpenSearch می‌تواند انتخاب مناسب‌تری باشد.

میزبان محیط شما نیز مهم است. اگر از سرویس‌های AWS استفاده می‌کنید، OpenSearch ممکن است ادغام و مدیریت روان‌تری را ارائه دهد. برعکس، اگر از ارائه‌دهنده ابری دیگری استفاده می‌کنید یا موتور جستجوی خود را به صورت محلی میزبانی می‌کنید، Elasticsearch ممکن است انعطاف‌پذیری بیشتری را فراهم کند.

در نهایت، ارزش دارد که به جهت آینده هر دو پروژه توجه کنید. Elasticsearch همچنان پیشرو جهانی در جستجوی سازمانی است و سابقه طولانی در نوآوری در این فضا دارد. OpenSearch Amazon، با جامعه محدودتر و تمرکز کمتر از سوی حامیان شرکتی، احتمالاً در نوآوری آینده نسبت به Elasticsearch عقب‌تر خواهد بود.

مفاهیم کلیدی OpenSearch :

موتور جستجو : OpenSearch چگونه جستجو را انجام می‌دهد؟

موتور جستجو در OpenSearch به گونه‌ای طراحی شده است که جستجوهای پیچیده و توزیع‌شده را با کارایی بالا انجام دهد. OpenSearch از ساختارهای داده‌ای پیشرفته مانند شاخص‌ها (indices) و شارد‌ها (shards) برای سازماندهی و مدیریت داده‌ها استفاده می‌کند. شاخص‌ها مجموعه‌ای از اسناد (documents) هستند که اطلاعات در آن‌ها ذخیره می‌شود. هر شاخص به چندین شارد تقسیم می‌شود که به طور توزیع‌شده در نودهای مختلف یک خوشه (cluster) قرار می‌گیرند. این توزیع داده‌ها باعث افزایش کارایی و قابلیت دسترسی به داده‌ها می‌شود.

پرس و جوها (queries) در OpenSearch به صورت JSON تعریف می شوند و از یک زبان پرس و جوی قدرتمند و انعطاف پذیر پشتیبانی می کنند. این زبان به کاربران امکان می دهد تا جستجوهای متنی، عددی، برداری و ترکیبی از آنها را انجام دهند. موتور جستجو در OpenSearch از تکنیک های پیشرفته مانند تجزیه و تحلیل متنی، شاخص گذاری معکوس (inverted indexing) و الگوریتم های رتبه بندی برای بهینه سازی نتایج جستجو استفاده می کند.

تجزیه و تحلیل داده : OpenSearch چه نوع تجزیه و تحلیل داده ای را ارائه می دهد؟

OpenSearch از قابلیت های پیشرفته ای برای تجزیه و تحلیل داده ها پشتیبانی می کند. این قابلیت ها شامل موارد زیر می شود:

تجزیه و تحلیل تجمیعی (Aggregations) : این قابلیت به کاربران امکان می دهد تا عملیات های تجمیعی مانند میانگین، ماکسیمم، مینیمم، جمع، و غیره را بر روی داده ها انجام دهند و اطلاعات کلی و جمع بندی شده ای از داده ها به دست آورند.

پایپ لاین های تحلیل (Analysis Pipelines) : مجموعه ای از مراحل تجزیه و تحلیل که به صورت متوالی بر روی داده ها اعمال می شوند و به کاربران امکان می دهند تا فرایندهای پیچیده تحلیل داده ها را طراحی و اجرا کنند.

شاخص های رول آپ (Rollup Indices) : شاخص هایی که داده های قدیمی را به صورت تجمیعی ذخیره می کنند تا حجم داده ها کاهش یابد و عملکرد بهبود یابد.

تجزیه و تحلیل زمانی (Time-series Analysis) : قابلیت هایی برای تحلیل داده های زمانی و ساخت داشبوردهای تعاملی برای مانیتورینگ و تحلیل داده های بلادرنگ.

مقیاس پذیری : OpenSearch چگونه مقیاس بندی می شود؟

OpenSearch با استفاده از معماری توزیع شده خود، قابلیت مقیاس پذیری بالایی دارد. این معماری به کاربران امکان می دهد تا داده ها را به صورت توزیع شده در چندین نود و شارد ذخیره و مدیریت کنند. مقیاس پذیری در OpenSearch از طریق موارد زیر تحقق می یابد:

شاردها (Shards): هر شاخص به چندین شارد تقسیم می‌شود و این شاردها به صورت توزیع شده در نودهای مختلف قرار می‌گیرند. این تقسیم‌بندی داده‌ها به کاربران امکان می‌دهد تا بار کاری را بین نودها توزیع کرده و عملکرد را بهبود بخشند.

افزودن نودهای جدید: کاربران می‌توانند به راحتی نودهای جدید به خوشه اضافه کنند تا ظرفیت ذخیره‌سازی و پردازش داده‌ها افزایش یابد. این قابلیت به خوشه امکان می‌دهد تا با افزایش حجم داده‌ها و بار کاری به صورت مقیاس‌پذیر عمل کند.

توزیع بار کاری: OpenSearch به صورت خودکار بار کاری را بین نودها توزیع می‌کند و اطمینان حاصل می‌کند که هیچ نودی بیش از حد بار نگیرد. این توزیع بار کاری باعث افزایش کارایی و کاهش زمان پاسخ می‌شود.

قابلیت استفاده : OpenSearch چقدر کاربرپسند است؟

OpenSearch به دلیل طراحی کاربرپسند و قابلیت‌های فراوان خود، یک ابزار بسیار محبوب و مورد استقبال در میان کاربران است. برخی از ویژگی‌هایی که OpenSearch را کاربرپسند می‌سازند عبارتند از:

داشبوردهای تعاملی (Interactive Dashboards): OpenSearch Dashboards به کاربران امکان می‌دهد تا داشبوردهای تعاملی و قابل سفارشی‌سازی ایجاد کنند و داده‌های خود را به صورت گرافیکی مشاهده و تحلیل کنند.

رابط کاربری گرافیکی (GUI): OpenSearch دارای یک رابط کاربری گرافیکی است که به کاربران امکان می‌دهد تا به راحتی شاخص‌ها، اسناد، و نودها را مدیریت کنند و پرس‌وجوهای خود را انجام دهند.

مستندات جامع: OpenSearch دارای مستندات جامع و کامل است که به کاربران کمک می‌کند تا با مفاهیم و قابلیت‌های مختلف آن آشنا شوند و به راحتی از آن استفاده کنند.

پلاگین‌ها و اکستنشن‌ها: کاربران می‌توانند از پلاگین‌ها و اکستنشن‌های مختلف برای گسترش قابلیت‌های OpenSearch استفاده کنند و آن را بر اساس نیازهای خاص خود سفارشی‌سازی کنند.

متن‌باز : OpenSearch چه معنای متن‌باز بودن دارد؟

OpenSearch به عنوان یک پروژه متن‌باز تحت لایسنس Apache 2.0 عرضه شده است. متن‌باز بودن

OpenSearch به معنای موارد زیر است:

دسترسی به کد منبع: کاربران و توسعه‌دهندگان می‌توانند به کد منبع **OpenSearch** دسترسی داشته باشند و آن را بررسی، تغییر، و بهبود دهند.

مشارکت جامعه: **OpenSearch** توسط یک جامعه فعال و پرجنب و جوش از توسعه‌دهندگان و کاربران حمایت می‌شود. این جامعه به توسعه و بهبود مستمر پروژه کمک می‌کند و تجربیات و دانش خود را با دیگران به اشتراک می‌گذارد.

استقلال از فروشنده: متن‌باز بودن **OpenSearch** به کاربران امکان می‌دهد تا از وابستگی به فروشندگان خاص خودداری کنند و به راحتی آن را بر روی زیرساخت‌های مختلف اجرا کنند.

شفافیت و اعتماد: متن‌باز بودن به معنای شفافیت در توسعه و به‌روزرسانی پروژه است. کاربران می‌توانند به راحتی تغییرات و به‌روزرسانی‌ها را دنبال کنند و از امنیت و پایداری سیستم اطمینان حاصل کنند.

متن‌باز بودن **OpenSearch** نه تنها باعث افزایش اعتماد کاربران می‌شود، بلکه به توسعه‌دهندگان امکان می‌دهد تا با همکاری و مشارکت در این پروژه، به بهبود و ارتقاء آن کمک کنند.

قابلیت‌های کلیدی:

۱. موتور جستجو:

OpenSearch قابلیت انجام عملیات جستجو را در انواع برنامه‌ها و وبسایت‌ها فراهم می‌کند. این پلتفرم با فهرست‌بندی و پرس‌وجوی حجم بزرگی از داده‌های ساختار یافته و غیرساختاری، نتایج جستجوی سریع و مرتبطی را ارائه می‌دهد.

۲. تجزیه و تحلیل لاگ و رویداد:

OpenSearch به طور معمول برای تجزیه و تحلیل لاگ و رویداد استفاده می‌شود. این امکان را فراهم می‌کند که لاگ‌ها از منابع مختلف مانند سرورها، برنامه‌ها و دستگاه‌های شبکه دریافت شده و سپس کاربران بتوانند بر روی داده‌های لاگ جستجو کنند، آن‌ها را تحلیل کرده و نتایج را به صورت گرافیکی نمایش دهند تا مشکلات را برطرف کنند، عملکرد سیستم را نظارت کنند و روندها یا نقص‌ها را شناسایی کنند.

۳. تحلیل داده‌های لحظه‌ای:

OpenSearch از تحلیل داده‌های لحظه‌ای پشتیبانی می‌کند با ارائه امکان فهرست‌بندی و پرس‌وجوی نزدیک به لحظه. این پلتفرم به سازمان‌ها اجازه می‌دهد تا منابع داده جریانی مانند دستگاه‌های **IoT**، پیام‌های رسانه‌های اجتماعی و شبکه‌های حسگر را برای به دست آوردن بینش و اتخاذ تصمیمات مبتنی بر داده در لحظه تحلیل کنند.

۴. هوش تجاری (BI):

OpenSearch می‌تواند به عنوان پشتیبانی برای برنامه‌های هوش تجاری (BI) عمل کند و به کاربران امکان انجام پرس‌وجوهای اد-هاک، تولید گزارشات و ایجاد نمودارهای بر اساس مجموعه داده‌های بزرگ را فراهم کند. این پلتفرم عملیات تجمیع، فیلتر و مرتب‌سازی پیچیده مورد نیاز برای موارد کاربردی هوش تجاری و تحلیل داده را پشتیبانی می‌کند.

۵. نظارت و قابلیت مشاهده:

OpenSearch برای اهداف نظارت و قابلیت مشاهده در سیستم‌های توزیع شده و محیط‌های ابری استفاده می‌شود. این پلتفرم می‌تواند معیارها، ردیابی‌ها و سایر داده‌های تلمتری را جمع‌آوری و تحلیل کند تا نظارت بر سلامت، عملکرد و دسترسی‌پذیری برنامه‌ها، خدمات و مؤلفه‌های زیرساخت انجام شود.

۶. مدیریت امنیت و رویدادهای امنیتی (SIEM):

OpenSearch می‌تواند برای تحلیل امنیتی و شناسایی تهدیدات امنیتی در عملیات سایبری استفاده شود. با تجزیه و تحلیل لاگ‌ها و رویدادهای امنیتی از منابع مختلف، به سازمان‌ها کمک می‌کند تا تهدیدات امنیتی را شناسایی و پاسخگویی به آن‌ها را انجام دهند، تحقیقات پس از حوادث را انجام دهند و وضعیت امنیتی خود را بهبود بخشند.

۷. قابلیت جستجو و فیلترینگ پیشرفته:

OpenSearch امکانات جستجوی پیشرفته را فراهم می‌کند از جمله:

Open source search: امکان جستجو در متن اسناد بر اساس کلمات کلیدی و عبارات.

Location search: امکان جستجو بر اساس موقعیت جغرافیایی یا مرزها.

Vector search: امکان جستجو بر اساس وکتورهای تعبیه شده

۸. قابلیت انعطاف پذیری و مقیاس پذیری:

قابلیت مقیاس‌پذیری افقی OpenSearch: به طور افقی قابلیت مقیاس‌پذیری را فراهم می‌کند که به کاربران اجازه می‌دهد به راحتی خوشه‌های جستجو خود را با اضافه کردن نودهای بیشتر افزایش دهند. این امر اطمینان می‌دهد که زیرساخت جستجو توانایی مدیریت حجم بزرگی از داده و ترافیک را دارا باشد.

قابلیت دسترسی بالا: OpenSearch قابلیت‌های دسترسی بالا را فراهم می‌کند از جمله تکرار داده و خودکار فیلورور برای اطمینان از اینکه خدمات جستجو حتی در صورت شکست یا مشکل در نودها یا مسائل شبکه قابل دسترسی باقی می‌مانند.

۹. امنیت و کنترل دسترسی:

OpenSearch امکانات گسترده‌ای برای امنیت داده‌ها و کنترل دسترسی فراهم می‌کند. این پلتفرم از روش‌های رمزنگاری داده‌ها، مدیریت هویت و دسترسی، مانیتورینگ فعالیت‌ها و ردیابی دسترسی استفاده می‌کند تا امنیت داده‌ها و منابع جستجو را تضمین کند. از طریق کنترل دسترسی به داده‌ها و سرویس‌های مرتبط، OpenSearch امکان اجرای استراتژی‌های امنیتی و حفاظت از داده‌ها را فراهم می‌کند.

۱۰. ادغام با سیستم‌های موجود:

OpenSearch امکان ادغام با سیستم‌های موجود در سازمان را فراهم می‌کند. این پلتفرم از API‌های استاندارد و قابل توسعه برای ارتباط با سیستم‌های دیگر استفاده می‌کند، این امر به سازمان‌ها کمک می‌کند تا OpenSearch را به سادگی با سیستم‌ها، ابزارها و فرآیندهای خود ادغام کنند و از امکانات آن در کنار سیستم‌های موجود استفاده کنند.

۱۱. پشتیبانی از استانداردهای صنعتی:

OpenSearch پشتیبانی از استانداردهای صنعتی را فراهم می‌کند که به سازمان‌ها کمک می‌کند تا با استفاده از پروتکل‌ها و فرمت‌های استاندارد، اطلاعات را بدون ایجاد تغییرات یا مشکلات سازگاری میان سیستم‌ها به اشتراک بگذارند. این پشتیبانی از استانداردهای صنعتی از جمله RESTful API، JSON و XML است.

۱۲. پشتیبانی از محیط‌های چندفرهنگی:

OpenSearch قابلیت پشتیبانی از محیط‌های چندفرهنگی را دارد که به کاربران از مختلف زبان‌ها و فرهنگ‌ها امکان استفاده از این پلتفرم را می‌دهد. این امکانات شامل پشتیبانی از چندین زبان برنامه‌نویسی، امکان ترجمه متون و رابط‌های کاربری چندزبانه، و قابلیت تطبیق با نیازهای محلی می‌شود.

۱۳. پشتیبانی از متن‌باز بودن:

OpenSearch به عنوان یک پروژه متن‌باز توسط جامعه‌ی توسعه‌دهنده‌ها حمایت می‌شود. این امر به افراد و سازمان‌ها امکان می‌دهد که به راحتی کدهای OpenSearch را بررسی، تغییر و بهبود دهند و از قابلیت‌های آن بهره‌برند.

۱۴. پشتیبانی از توسعه پذیری و انعطاف پذیری:

OpenSearch از یک معماری قابل توسعه و انعطاف‌پذیر بهره می‌برد که به توسعه‌دهندگان امکان می‌دهد تا بر اساس نیازهای خود این پلتفرم را گسترش دهند و به شکلی مناسب با فرآیندها و الگوهای کسب‌وکار خود تنظیم کنند.

مهاجرت از Elasticsearch OSS به OpenSearch : (اطلاعات بیشتر)

مهاجرت از Elasticsearch OSS به OpenSearch فرآیندی است که به شما امکان می دهد از مزایای OpenSearch، مانند مقیاس پذیری بهتر، امنیت و قابلیت های تحلیلی پیشرفته، بهره مند شوید. با این حال، این فرآیند می تواند پیچیده باشد، به خصوص اگر با حجم زیادی از داده ها یا خوشه های Elasticsearch پیچیده سروکار دارید. در این راهنمای جامع، ما سه روش اصلی مهاجرت را به تفصیل شرح می دهیم:

۱. مهاجرت با استفاده از Snapshot :

- این روش شامل گرفتن snapshot از داده های Elasticsearch OSS شما و سپس بازیابی آن در یک خوشه OpenSearch جدید است.
- این روش ساده ترین روش است، اما می تواند باعث خرابی شود، زیرا کل خوشه Elasticsearch باید برای انجام snapshot متوقف شود.

۲. ارتقاء نورد:

- این روش شامل ارتقاء هر گره در خوشه Elasticsearch شما به OpenSearch به صورت جداگانه است.
- این روش مزیت عدم ایجاد خرابی را دارد، اما می تواند زمان بر باشد و به مدیریت دقیق نیاز داشته باشد.

۳. جایگزینی گره:

- این روش شامل جایگزینی هر گره در خوشه Elasticsearch شما با یک گره OpenSearch جدید است.
- این روش سریع و کارآمد است، اما می تواند پرهزینه باشد، به خصوص اگر خوشه بزرگی داشته باشید.

مدیریت ایندکس در OpenSearch : (اطلاعات بیشتر)

فرایند ایندکس کردن داده ها در OpenSearch با استفاده از API های REST انجام می شود. دو API کلیدی برای این منظور وجود دارد:

- **API ایندکس:** این API برای اضافه کردن تک به تک اسناد (دیتا) به کار می رود، زمانی که داده ها به صورت پیوسته و با حجم کم وارد می شوند، مانند ثبت سفارش مشتریان در یک کسب و کار کوچک.

- **API با عملکرد حجمی (bulk):** این API برای حجم بالایی از اسناد و زمانی که جریان ورود داده ها منظم نیست، مانند روزرسانی هفتگی وب سایت بازاریابی، کاربرد دارد. برای تعداد زیادی از اسناد، جمع کردن درخواست ها و استفاده از API با عملکرد حجمی، عملکرد بهتری را ارائه می دهد. با این حال، اگر اسناد شما بسیار حجیم هستند، ممکن است نیاز به ایندکس کردن آنها به صورت جداگانه داشته باشید.

ملاحظات کلیدی در هنگام ایندکس کردن اسناد:

- اندازه شناسه (ID) هر سند هنگام ایندکس کردن باید حداکثر ۵۱۲ بایت باشد.

مقدمه ای بر ایندکس کردن

قبل از اینکه بتوانید داده ها را جستجو کنید، باید آنها را ایندکس کنید. ایندکس کردن روشی است که موتورهای جستجو برای سازماندهی داده ها جهت بازیابی سریع از آنها استفاده می کنند. خروجی این فرآیند، ساختاری به نام ایندکس است. در OpenSearch، واحد اصلی داده یک سند JSON است OpenSearch. هر سند را با استفاده از یک شناسه (ID) منحصر به فرد در داخل یک ایندکس شناسایی می کند.

برای نمونه یک درخواست به API ایندکس به شکل زیر است:

```
PUT <index>/_doc/<id>
{ "A JSON": "document" }
```

درخواست به bulk API کمی متفاوت به نظر می رسد، زیرا شما شاخص و شناسه را در داده های انبوه مشخص می کنید:

```
POST _bulk
{ "index": { "_index": "<index>", "_id": "<id>" } }
{ "A JSON": "document" }
```

داده های انبوه باید با قالب خاصی مطابقت داشته باشند، که به یک کاراکتر خط جدید (\n) در انتهای هر خط، از جمله آخرین خط، نیاز دارد. برای مثال فرمت اصلی به شکل زیر است:

```
Action and metadata\n
Optional document\n
Action and metadata\n
Optional document\n
```


همچنین برای فهرست کردن داده های انبوه با استفاده از دستور `curl`، باید به پوشه ای که فایل خود را در آن ذخیره کرده ایم برویم و دستور زیر را اجرا کنیم:

```
curl -H "Content-Type: application/x-ndjson" -POST
https://localhost:9200/data/_bulk -u 'admin:admin' --insecure --data-binary
"@data.json"
```

اگر یکی از عملیات انجام شده در API با عملکرد حجمی (`_bulk`) با خطا مواجه شود، `OpenSearch` به اجرای سایر عملیات ادامه می دهد. برای بررسی موارد خطا، باید آرایه «items» در پاسخ را بررسی کنید. ترتیب قرارگیری موارد (items) در این آرایه با ترتیب عملیات مشخص شده در درخواست مطابقت دارد.

`OpenSearch` زمانی که سندی را به ایندکسی که هنوز وجود ندارد اضافه می کنید، به صورت خودکار آن ایندکس را ایجاد می کند. همچنین، در صورتی که در درخواست خود برای سند، شناسه (ID) خاصی تعیین نکرده باشید، `OpenSearch` به صورت خودکار یک شناسه (ID) منحصر به فرد برای آن سند تولید می کند.

این مثال ساده به طور خودکار فهرست فیلم ها را ایجاد می کند، سند را نمایه می کند و یک شناسه منحصر به فرد به آن اختصاص می دهد:

```
POST movies/_doc
{ "title": "Spirited Away" }
```

معایب تولید خودکار شناسه (ID)

ایجاد خودکار شناسه (ID) توسط `OpenSearch` یک نقطه ضعف آشکار دارد: از آنجایی که درخواست ایندکس کردن، شناسه ای برای سند مشخص نکرده است، به سادگی نمی توانید بعداً آن سند را بروزرسانی کنید. همچنین، اگر این درخواست را ۱۰ بار اجرا کنید، `OpenSearch` این سند را به عنوان ۱۰ سند مجزا با شناسه های (ID) منحصر به فرد ایندکس می کند.

برای اینکه به صورت دستی شناسه (ID) را روی ۱ تنظیم کنید، از درخواست زیر استفاده کنید (توجه داشته باشید که به جای POST از PUT استفاده شده است):

```
PUT movies/_doc/1
{ "title": "Spirited Away" }
```

از آنجایی که باید یک شناسه مشخص کنید، اگر این دستور را ۱۰ بار اجرا کنید، باز هم فقط یک سند ایندکس شده با فیلد `_version` به ۱۰ افزایش یافته است. ایندکس‌ها به‌طور پیش‌فرض روی یک قطعه اولیه و یک `replica` هستند. اگر می‌خواهید تنظیمات غیر پیش‌فرض را مشخص کنید، قبل از افزودن اسناد، فهرست را ایجاد کنید:

```
PUT more-movies
{ "settings": { "number_of_shards": 6, "number_of_replicas": 2 } }
```

:Read data

پس از ایندکس کردن یک سند، می‌توانید با ارسال یک درخواست `GET` به همان نقطه پایانی که برای نمایه سازی استفاده کرده اید، آن را بازیابی کنید:

```
GET movies/_doc/1

{
  "_index" : "movies",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "_seq_no" : 0,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "title" : "Spirited Away"
  }
}
```

بروزرسانی اطلاعات:

برای به روز رسانی فیلدهای موجود یا افزودن فیلدهای جدید، یک درخواست `POST` به عملیات `update` با تغییرات خود در یک شیء `doc` ارسال کنید:

```
POST movies/_update/1
{
  "doc": {
    "title": "Castle in the Sky",
    "genre": ["Animation", "Fantasy"]
  }
}
```

به فیلد عنوان به روز شده و فیلد ژانر جدید توجه کنید:

```
GET movies/_doc/1

{
  "_index" : "movies",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 2,
  "_seq_no" : 1,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "title" : "Castle in the Sky",
    "genre" : [
      "Animation",
      "Fantasy"
    ]
  }
}
```

این سند همچنین دارای یک فیلد `_version` افزایش یافته است. از این فیلد برای پیگیری تعداد دفعات به‌روزرسانی یک سند استفاده کنید.

درخواست‌های `POST` به‌روزرسانی‌های جزئی اسناد را انجام می‌دهند. برای جایگزینی کلی یک سند، از یک درخواست `PUT` استفاده کنید:

```
PUT movies/_doc/1
{
  "title": "Spirited Away"
}
```

حذف داده ها:

برای حذف یک سند از یک فهرست، از درخواست `DELETE` استفاده کنید:

```
DELETE movies/_doc/1
```

قالب های ایندکس:

قالب های ایندکس به شما این امکان را می دهند که ایندکس های جدید را با نگاشت ها (mappings) و تنظیمات از پیش تعریف شده، راه اندازی کنید. برای مثال، اگر به طور مداوم داده های لاگ (log) را ایندکس می کنید، می توانید یک قالب ایندکس تعریف کنید تا همه این ایندکس ها تعداد یکسانی Shard و Replica داشته باشند.

ایجاد یک قالب

برای ایجاد یک قالب ایندکس، از درخواست PUT یا POST استفاده کنید:

```
PUT _index_template/<template name>
POST _index_template/<template name>
```

نام مستعار شاخص (Index aliases) :

یک Alias در OpenSearch، یک نام مجازی برای ایندکس است که می تواند به یک یا چند ایندکس واقعی اشاره کند. تصور کنید داده های شما در چندین ایندکس پراکنده شده اند. به جای اینکه دائماً دنبال کنید که باید کدام ایندکس ها را برای جستجو بررسی کنید، می توانید یک Alias ایجاد کرده و به جای آن، آن Alias را مورد پرس و جو قرار دهید.

به عنوان مثال، فرض کنید لاگ ها را بر اساس ماه در ایندکس های جداگانه ذخیره می کنید و اغلب برای لاگ های دو ماه گذشته کوئری ارسال می کنید. در این سناریو می توانید یک Alias با نام last_2_months ایجاد کنید و این Alias را طوری تنظیم کنید که به ایندکس های حاوی لاگ های دو ماه گذشته اشاره کند.

از آنجایی که می توانید ایندکس هایی را که یک Alias به آنها اشاره می کند در هر زمان تغییر دهید، استفاده از Alias ها در برنامه های کاربردی برای اشاره به ایندکس ها به شما این امکان را می دهد تا بدون هیچ گونه downtime ای داده های خود را مجدداً ایندکس کنید.

جریان های داده (Data Streams) :

اگر به طور مداوم در حال دریافت داده های سری زمانی مانند لاگ ها، رویدادها و معیارها در OpenSearch هستید، به احتمال زیاد در سناریویی قرار دارید که تعداد اسناد به سرعت افزایش می یابد و نیازی به به روز رسانی اسناد قدیمی ندارید.

یک گردش کار معمولی برای مدیریت داده های سری زمانی شامل چندین مرحله است، مانند ایجاد یک alias چرخشی برای ایندکس، تعریف یک ایندکس نوشتاری و تعریف نگاشت ها و تنظیمات مشترک برای ایندکس های پشتیبان.

جریان های داده این فرآیند را ساده می کنند و تنظیمی را اعمال می کنند که به بهترین نحو برای داده های سری زمانی مناسب است، مانند اینکه عمدتاً برای داده های الحاقی طراحی شده اند و اطمینان حاصل می کنند که هر سند دارای یک فیلد زمان بندی (timestamp) باشد.

یک جریان داده از نظر داخلی از چندین ایندکس پشتیبان تشکیل شده است. درخواست های جستجو به همه ایندکس های پشتیبان هدایت می شوند، در حالی که درخواست های ایندکس کردن به آخرین ایندکس نوشتاری هدایت می شوند. سیاست های مدیریت چرخه عمر ایندکس (ISM) به شما امکان می دهند به طور خودکار چرخش یا حذف ایندکس ها را مدیریت کنید.

مدیریت وضعیت ایندکس (Index State Management) :

اگر با داده های سری زمانی سروکار دارید، به احتمال زیاد اولویت شما داده های جدیدتر نسبت به داده های قدیمی تر است. شما ممکن است به طور دوره ای عملیات خاصی را روی ایندکس های قدیمی انجام دهید، مانند کاهش تعداد Replica یا حذف آنها.

مدیریت وضعیت ایندکس (ISM) یک افزونه است که به شما امکان می دهد این عملیات اداری دوره ای را با فعال کردن آنها بر اساس تغییرات در سن ایندکس، اندازه ایندکس یا تعداد اسناد، به صورت خودکار انجام دهید. با استفاده از افزونه ISM، می توانید «سیاست هایی» را تعریف کنید که به صورت خودکار چرخش ایندکس (rollover) یا حذف آنها را مطابق با سناریوی کاری شما مدیریت کنند.

به عنوان مثال، می توانید یک سیاست تعریف کنید که ایندکس شما را پس از ۳۰ روز به حالت فقط خواندنی (read-only) منتقل کند و سپس پس از دوره تعیین شده ۹۰ روزه آن را حذف کند. همچنین می توانید این سیاست را برای ارسال پیام هشدار به شما هنگام حذف ایندکس تنظیم کنید.

ممکن است بخواهید پس از مدت زمان معینی یک چرخش ایندکس انجام دهید یا در ساعات غیر اوج، عملیات «ادغام اجباری» (force_merge) را روی یک ایندکس اجرا کنید تا عملکرد جستجو را در ساعات اوج بهبود بخشید.

تبدیل ایندکس (Transform Jobs) :

در حالی که عملیات خلاصه سازی ایندکس (rollup) به شما امکان می دهد با فشرده سازی داده های قدیمی در ایندکس های خلاصه شده، حجم داده را کاهش دهید، عملیات تبدیل (transform) به شما این امکان را می دهد تا یک نمای کلی و خلاصه شده متفاوت از داده های خود، بر اساس فیلدهای خاص ایجاد کنید. بدین ترتیب می توانید داده ها را به روش های مختلف تجسم یا تحلیل کنید.

به عنوان مثال، فرض کنید داده هایی در مورد خطوط هوایی دارید که در چندین فیلد و دسته پراکنده شده اند. شما می خواهید خلاصه ای از داده ها را مشاهده کنید که بر اساس ایرلاین، سه ماهه و سپس قیمت سازماندهی شده باشد. می

توانید از یک عملیات تبدیل برای ایجاد یک ایندکس جدید و خلاصه شده استفاده کنید که بر اساس آن دسته بندی های خاص سازماندهی شده است.

روش های انجام عملیات تبدیل

شما می توانید از دو روش برای انجام عملیات تبدیل استفاده کنید:

- **استفاده از رابط کاربری: OpenSearch Dashboards** در این روش می توانید ایندکسی را که می خواهید تبدیل کنید و هر فیلتر داده اختیاری که می خواهید برای فیلتر کردن ایندکس اصلی استفاده کنید، مشخص کنید. سپس فیلدهایی را که می خواهید تبدیل کنید و توابع تجمع (aggregations) مورد نظر برای تبدیل را انتخاب کنید. در نهایت، یک زمانبندی برای اجرای این عملیات تبدیل تعریف کنید.
- **استفاده از API تبدیل: (Transforms)** با استفاده از این روش می توانید تمام جزئیات مربوط به عملیات تبدیل خود را مشخص کنید. این جزئیات شامل ایندکسی که می خواهید تبدیل کنید، گروه های مقصدی که می خواهید ایندکس تبدیل شده داشته باشد، هرگونه تابع جمعی که می خواهید برای گروه بندی ستون ها استفاده کنید و زمانبندی برای اجرای این عملیات تبدیل می شود.

OpenSearch Dashboards در مقابل API تبدیل :

OpenSearch Dashboards یک نمای کلی از عملیات تبدیل هایی که ایجاد کرده اید و اطلاعات مرتبط با آنها، مانند ایندکس های مرتبط و وضعیت عملیات را ارائه می دهد. شما می توانید جزئیات و انتخاب های عملیات تبدیل خود را قبل از ایجاد بررسی و ویرایش کنید، و حتی می توانید هنگام انتخاب فیلدهایی که می خواهید تبدیل کنید، پیش نمایشی از داده های ایندکس تبدیل شده را مشاهده کنید.

با این حال، شما همچنین می توانید از REST API برای ایجاد عملیات تبدیل و پیش نمایش نتایج آنها استفاده کنید، اما باید تمام تنظیمات و پارامترهای لازم را بدانید تا آنها را به عنوان بخشی از بدنه درخواست HTTP ارسال کنید. ارسال تنظیمات عملیات تبدیل شما به صورت اسکریپت های JSON قابلیت حمل بیشتری را برای شما فراهم می کند و به شما امکان می دهد عملیات تبدیل خود را به اشتراک بگذارید و تکرار کنید، که انجام این کار با استفاده از OpenSearch Dashboards دشوارتر است. تصمیم گیری در مورد اینکه از کدام روش برای ایجاد عملیات تبدیل استفاده کنید، بر اساس موارد استفاده شما انجام می شود.

خلاصه سازی ایندکس (Index Rollups)

داده های سری زمانی به مرور زمان هزینه ذخیره سازی را افزایش می دهند، سلامت خوشه را تحت فشار قرار می دهند و سرعت انجام عملیات جمع (aggregations) را کاهش می دهند. خلاصه سازی ایندکس به شما این امکان را می دهد که به طور دوره ای با فشرده سازی داده های قدیمی در ایندکس های خلاصه شده، جزئیات داده ها را کاهش دهید.

در خلاصه سازی ایندکس، شما فیلدهای مورد نظر خود را انتخاب می کنید و با استفاده از این عملیات، یک ایندکس جدید ایجاد می کنید که تنها آن فیلدها را در بازه های زمانی بزرگتر خلاصه می کند. به این ترتیب می توانید ماه ها یا سال ها داده تاریخی را با همان عملکرد جستجو و با هزینه ای بسیار کمتر ذخیره کنید.

به عنوان مثال، فرض کنید داده های مربوط به مصرف CPU را هر پنج ثانیه جمع آوری می کنید و آنها را در یک گره داغ (Hot Node) ذخیره می کنید. به جای انتقال داده های قدیمی تر به یک گره گرم (Warm Node) فقط خواندنی، می توانید این داده ها را با خلاصه سازی یا فشرده سازی، فقط با میانگین مصرف CPU روزانه یا با کاهش ۱۰ درصدی فاصله زمانی آنها در هر هفته، خلاصه کنید.

روش های انجام خلاصه سازی ایندکس

شما می توانید از سه روش برای انجام خلاصه سازی ایندکس استفاده کنید:

- **استفاده از API خلاصه سازی ایندکس:** این روش برای یک عملیات خلاصه سازی ایندکس در صورت نیاز (On-Demand) بر روی ایندکسی که در حال دریافت مداوم داده نیست، مانند یک ایندکس چرخشی (Rolled-Over)، کاربرد دارد. به عنوان مثال، می توانید یک عملیات خلاصه سازی ایندکس را برای کاهش داده های جمع آوری شده در بازه زمانی پنج دقیقه به میانگین هفتگی برای تحلیل روند، انجام دهید.
- **استفاده از رابط کاربری OpenSearch Dashboards:** با استفاده از این رابط کاربری می توانید یک عملیات خلاصه سازی ایندکس ایجاد کنید که بر اساس یک زمانبندی مشخص اجرا شود. همچنین می توانید آن را برای خلاصه سازی ایندکس های خود در حین دریافت مداوم داده، تنظیم کنید. به عنوان مثال، می توانید به طور مداوم ایندکس های Logstash را از بازه زمانی پنج ثانیه به بازه زمانی یک ساعت خلاصه کنید.
- **تعریف عملیات خلاصه سازی ایندکس به عنوان یک اکشن در مدیریت وضعیت ایندکس (ISM):** این روش به شما امکان می دهد تا یک ایندکس را پس از یک رویداد خاص مانند چرخش ایندکس، رسیدن سن ایندکس به یک نقطه خاص، تبدیل شدن ایندکس به فقط خواندنی و غیره، خلاصه کنید. همچنین می توانید عملیات چرخش و خلاصه سازی ایندکس را به صورت متوالی اجرا کنید، به این ترتیب که ابتدا عملیات چرخش ایندکس جاری را به یک گره گرم منتقل کند و سپس عملیات خلاصه سازی ایندکس یک ایندکس جدید با داده های خلاصه شده در گره داغ ایجاد کند.

امنیت مدیریت ایندکس

پلاگین امنیتی (Security Plugin) به شما این امکان را می دهد تا با مدیریت ایندکس، دسترسی کاربران غیرمدیر را به اقدامات خاصی محدود کنید. به عنوان مثال، ممکن است بخواهید امنیت خود را طوری تنظیم کنید که گروهی از کاربران فقط بتوانند سیاست های مدیریت وضعیت ایندکس (ISM) را بخوانند، در حالی که گروهی دیگر بتوانند این سیاست ها را ایجاد، حذف یا تغییر دهند.

تمام داده های مدیریت ایندکس به عنوان ایندکس های سیستم محافظت می شوند و تنها یک مدیر ارشد (Super Admin) یا یک مدیر با گواهی امنیتی لایه انتقال (TLS) می تواند به ایندکس های سیستم دسترسی داشته باشد. برای اطلاعات بیشتر به بخش «ایندکس های سیستم» مراجعه کنید.

OpenSearch Dashboards: (اطلاعات بیشتر)

OpenSearch Dashboards، رابط کاربری بصری OpenSearch است که به شما امکان می دهد داده های خود را به طور موثر تجسم و تحلیل کنید. این ابزار قدرتمند به شما کمک می کند تا از طریق نمودارها، گراف ها، جداول و سایر عناصر بصری، بینش عمیقی از داده های خود به دست آورید.

قابلیت های کلیدی OpenSearch Dashboards :

- **ایجاد داشبوردهای سفارشی:** داشبوردهایی را با چیدمان دلخواه خود و شامل انواع مختلف نمودارها، گراف ها، جداول و سایر عناصر بصری ایجاد کنید.
- **تجسم داده های تان:** از انواع مختلف نمودارها، گراف ها و جداول برای تجسم داده های خود به روشی بصری و قابل فهم استفاده کنید.
- **تحلیل داده ها:** با استفاده از فیلترها، جستجو و سایر ابزارهای تحلیل داده، داده های خود را عمیق تر کاوش کنید.
- **اشتراک گذاری داشبوردها:** داشبوردهای خود را با دیگران به اشتراک بگذارید تا آنها نیز بتوانند از بینش های شما بهره مند شوند.
- **افزونه ها:** با استفاده از افزونه ها، قابلیت های OpenSearch Dashboards را گسترش دهید.

مزایای استفاده از OpenSearch Dashboards :

- **بهبود درک از داده ها:** تجسم داده ها به صورت بصری به شما کمک می کند تا الگوها، روندها و ناهنجاری ها را به سرعت و به آسانی شناسایی کنید.
- **تصمیم گیری بهتر:** با داشتن بینش عمیق تر از داده های خود، می توانید تصمیمات آگاهانه تری بگیرید.
- **افزایش بهره وری OpenSearch Dashboards:** به شما کمک می کند تا به سرعت اطلاعات مورد نیاز خود را پیدا کنید و زمان خود را صرف تحلیل داده ها به جای جستجوی آنها کنید.
- **ارتباط بهتر:** داشبوردهای خود را با دیگران به اشتراک بگذارید تا دیدگاه ها و ایده های خود را به اشتراک بگذارید و همکاری موثرتری داشته باشید.

OpenSearch Dashboards برای چه کسانی مناسب است؟

OpenSearch Dashboards برای هر کسی که با داده ها سروکار دارد، مفید است. از تحلیلگران داده و دانشمندان داده گرفته تا مدیران و صاحبان مشاغل، همه می توانند از این ابزار برای به دست آوردن بینش از داده های خود و اتخاذ تصمیمات بهتر استفاده کنند.

شروع کار با OpenSearch Dashboards :

شروع کار با OpenSearch Dashboards آسان است. فقط کافی است OpenSearch را نصب و راه اندازی کنید و سپس به OpenSearch Dashboards در مرورگر وب خود دسترسی پیدا کنید. برای اطلاعات بیشتر، به مستندات OpenSearch Dashboards مراجعه کنید.

OpenSearch Dashboards، ابزاری قدرتمند و انعطاف پذیر برای تجسم و تحلیل داده ها است که می تواند به شما در به دست آوردن بینش از داده هایتان و اتخاذ تصمیمات بهتر کمک کند.

امنیت در OpenSearch (اطلاعات بیشتر)

امنیت در OpenSearch حول چهار ویژگی اصلی ساخته شده است که برای محافظت از داده ها و ردیابی فعالیت در یک خوشه با هم کار می کنند. به طور جداگانه، این ویژگی ها عبارتند از:

- رمزگذاری
- احراز هویت
- کنترل دسترسی
- ثبت حسابرسی و انطباق

آنها با هم استفاده می شوند و با قرار دادن آنها در پشت چندین لایه دفاعی و اعطا یا محدود کردن دسترسی به داده ها در سطوح مختلف در ساختار داده OpenSearch، حفاظت موثری از داده های حساس را فراهم می کنند. اکثر پیاده سازی ها از ترکیبی از گزینه ها برای این ویژگی ها برای رفع نیازهای امنیتی خاص استفاده می کنند.

ویژگی ها در یک نگاه

عناوین زیر توضیحاتی کلی از ویژگی هایی که امنیت را در OpenSearch تعریف می کنند ارائه می دهد.

رمزگذاری

رمزگذاری معمولاً به حفاظت از داده ها در حالت استراحت و انتقال می پردازد. OpenSearch Security مسئول مدیریت رمزگذاری در حین حمل و نقل است.

در حین انتقال، Security داده هایی را که به سمت، از و درون خوشه حرکت می کنند، رمزگذاری می کند. OpenSearch از پروتکل TLS استفاده می کند که هم رمزگذاری مشتری به گره (لایه REST) و هم رمزگذاری گره به گره (لایه انتقال) را پوشش می دهد. این ترکیب از رمزگذاری در حین حمل و نقل کمک می کند تا اطمینان حاصل شود که هر دو درخواست به OpenSearch و حرکت داده ها در میان گره های مختلف از دستکاری در امان هستند.

از سوی دیگر، رمزگذاری در حالت استراحت، از داده های ذخیره شده در خوشه، از جمله فهرست ها، گزارش ها، فایل های مبادله، عکس های فوری خودکار و همه داده های موجود در فهرست برنامه محافظت می کند. این نوع رمزگذاری توسط سیستم عامل در هر گره OpenSearch مدیریت می شود.

احراز هویت

احراز هویت برای تأیید هویت کاربران استفاده می شود و با تأیید اعتبار کاربر نهایی در برابر پیکربندی باطن کار می کند. این اعتبارنامه ها می تواند یک نام و رمز عبور ساده، یک رمز وب JSON یا یک گواهی TLS باشد. هنگامی که دامنه احراز هویت آن اعتبارنامه ها را از درخواست کاربر استخراج می کند، می تواند اعتبار آنها را در برابر باطن احراز هویت بررسی کند.

پشتیبان مورد استفاده برای اعتبارسنجی می تواند پایگاه داده داخلی داخلی OpenSearch باشد - که برای ذخیره پیکربندی های کاربر و نقش و رمزهای عبور هش شده استفاده می شود - یا یکی از طیف گسترده ای از پروتکل های شناسایی استاندارد صنعتی مانند LDAP، Active Directory، SAML، یا OpenID Connect. یک روش متداول این است که بیش از یک روش احراز هویت را به هم متصل کنید تا دفاع قوی تری در برابر دسترسی غیرمجاز ایجاد کنید. این ممکن است برای مثال شامل احراز هویت اولیه HTTP و به دنبال آن یک پیکربندی باطن باشد که پروتکل LDAP را مشخص می کند.

کنترل دسترسی

کنترل دسترسی (یا مجوز) به طور کلی شامل تخصیص انتخابی مجوزها به کاربران است که به آنها اجازه می دهد وظایف خاصی را انجام دهند، مانند پاک کردن حافظه پنهان برای یک فهرست خاص یا گرفتن عکس فوری از یک خوشه. با این حال، OpenSearch به جای اختصاص مجوزهای فردی مستقیماً به کاربران، این مجوزها را به نقش ها اختصاص می دهد و سپس نقش ها را برای کاربران ترسیم می کند. برای اطلاعات بیشتر در مورد تنظیم این روابط، به [کاربران و نقش ها](#) مراجعه کنید. بنابراین، نقش ها اقداماتی را که کاربران می توانند انجام دهند، از جمله داده هایی که می توانند بخوانند، تنظیمات خوشه ای که می توانند تغییر دهند، فهرست هایی که می توانند روی آن بنویسند و غیره را تعریف می کنند. نقش ها برای چندین کاربر قابل استفاده مجدد هستند و کاربران می توانند نقش های متعددی داشته باشند.

یکی دیگر از ویژگی‌های قابل توجه کنترل دسترسی در OpenSearch، توانایی اختصاص دسترسی کاربر از طریق سطوح افزایش جزئیات است. کنترل دسترسی ریز دانه (FGAC) به این معنی است که یک نقش می‌تواند مجوزهای کاربران را نه تنها در سطح خوشه بلکه در سطح فهرست، سطح سند و حتی سطح فیلد کنترل کند. به عنوان مثال، یک نقش ممکن است دسترسی کاربر را به برخی از مجوزهای سطح خوشه فراهم کند، اما در عین حال کاربر را از دسترسی به یک گروه مشخص از نمایه‌ها جلوگیری کند. به همین ترتیب، آن نقش ممکن است به انواع خاصی از اسناد دسترسی داشته باشد اما به انواع دیگر دسترسی نداشته باشد، یا حتی ممکن است شامل دسترسی به فیلدهای خاص در یک سند باشد، اما دسترسی به سایر زمینه‌های حساس را ممنوع کند. پوشاندن فیلد FGAC را با ارائه گزینه‌هایی برای پنهان کردن انواع خاصی از داده‌ها، مانند فهرستی از ایمیل‌ها، که هنوز هم می‌توانند جمع‌آوری شوند، اما برای یک نقش قابل مشاهده نیستند، گسترش می‌دهد.

ثبت حسابرسی و انطباق

در نهایت، ثبت حسابرسی و انطباق به مکانیسم‌هایی اشاره دارد که امکان ردیابی و تجزیه و تحلیل فعالیت در یک خوشه را فراهم می‌کند. این مهم پس از نقض داده‌ها (دسترسی غیرمجاز) یا زمانی که داده‌ها در معرض قرار گرفتن ناخواسته قرار می‌گیرند، مهم است، همانطور که ممکن است زمانی اتفاق بیفتد که داده‌ها در یک مکان ناامن آسیب پذیر باشند. با این حال، ثبت حسابرسی می‌تواند به همان اندازه ابزار ارزشمندی برای ارزیابی بارهای بیش از حد روی یک خوشه یا بررسی روندها برای یک کار معین باشد. این ویژگی به شما امکان می‌دهد تغییرات ایجاد شده را در هر نقطه از یک خوشه بررسی کنید و الگوهای دسترسی و درخواست‌های API را از همه نوع، اعم از معتبر یا نامعتبر، ردیابی کنید.

نحوه ثبت بایگانی OpenSearch در سطوح مختلف جزئیات قابل تنظیم است، و تعدادی گزینه برای محل ذخیره آن گزارش‌ها وجود دارد. ویژگی‌های انطباق همچنین تضمین می‌کنند که در صورت نیاز و زمانی که ممیزی انطباق مورد نیاز است، همه داده‌ها در دسترس هستند. در این مورد، ثبت گزارش می‌تواند خودکار شود تا روی داده‌هایی که مخصوصاً مربوط به آن الزامات انطباق است تمرکز کند.

سایر ویژگی‌ها و قابلیت‌ها

OpenSearch شامل ویژگی‌های دیگری است که زیرساخت امنیتی را تکمیل می‌کند.

داشبوردهای چند اجاره‌ای

یکی از این ویژگی‌ها، چند اجاره‌بندی داشبوردهای OpenSearch است. مستأجرها فضاهای کاری هستند که شامل تجسم‌ها، الگوهای فهرست و سایر اشیاء داشبورد می‌شوند. چند اجاره‌ای امکان اشتراک مستاجرین را در میان کاربران داشبوردها فراهم می‌کند و از نقش‌های OpenSearch برای مدیریت دسترسی مستاجرین و در دسترس قرار دادن ایمن آنها برای دیگران استفاده می‌کند.

جستجوی متقابل خوشه ای

یکی دیگر از ویژگی های قابل توجه جستجوی متقابل خوشه ای است. این ویژگی به کاربران امکان می دهد تا جستجوها را از یک گروه در یک خوشه در میان خوشه های دیگری که برای هماهنگی این نوع جستجو تنظیم شده اند انجام دهند. مانند سایر ویژگی ها، جستجوی متقابل خوشه ای توسط زیرساخت کنترل دسترسی OpenSearch پشتیبانی می شود که مجوزهایی را که کاربران برای کار با این ویژگی دارند، تعریف می کند.

تجزیه و تحلیل امنیتی

تجزیه و تحلیل امنیتی یک راه حل اطلاعات امنیتی و مدیریت رویداد (SIEM) برای OpenSearch است که برای بررسی، شناسایی، تجزیه و تحلیل و پاسخ به تهدیدهای امنیتی که می توانند موفقیت کسب و کارها و سازمان ها و عملیات آنلاین آنها را به خطر بیندازند، طراحی شده است. این تهدیدها شامل قرار گرفتن در معرض احتمالی داده های محرمانه، حملات سایبری و سایر رویدادهای امنیتی نامطلوب است. Security Analytics یک راه حل خارج از جعبه ارائه می دهد که به طور خودکار با هر توزیع OpenSearch نصب می شود. این شامل ابزارها و ویژگی های لازم برای تعریف پارامترهای تشخیص، تولید هشدارها و پاسخگویی موثر به تهدیدات بالقوه است.

منابع و اطلاعات

به عنوان بخشی از پروژه OpenSearch، آنالیز امنیتی در جامعه منبع باز وجود دارد و از بازخورد و مشارکت آن جامعه بهره می برد.

اجزا و مفاهیم

آنالیز امنیتی شامل تعدادی ابزار و ویژگی های اساسی برای عملکرد آن است. اجزای اصلی که افزونه را تشکیل می دهند در بخش های زیر خلاصه می شوند.

آشکارسازها

آشکارسازها اجزای اصلی هستند که برای شناسایی طیفی از تهدیدات امنیت سایبری مطابق با پایگاه دانش رو به رشد تاکتیک ها و تکنیک های دشمن که توسط سازمان [MITER ATT&CK](#) نگهداری می شود، پیکربندی شده اند. آشکارسازها از داده های گزارش برای ارزیابی رویدادهای رخ داده در سیستم استفاده می کنند. سپس مجموعه ای از قوانین امنیتی مشخص شده برای آشکارساز را اعمال می کنند و یافته های این رویدادها را تعیین می کنند.

انواع لاگ

[انواع گزارش ها](#) داده های مورد استفاده برای ارزیابی رویدادهای رخ داده در یک سیستم را فراهم می کنند. OpenSearch از چندین نوع گزارش پشتیبانی می کند و نگاشت های خارج از جعبه را برای رایج ترین منابع گزارش ارائه می دهد.

انواع گزارش ها در طول ایجاد آشکارسازها، از جمله مراحل نگاشت فیلدهای گزارش به آشکارساز، مشخص می شوند. Security Analytics همچنین به طور خودکار مجموعه ای از قوانین مناسب را بر اساس یک نوع گزارش خاص انتخاب می کند و آنها را برای آشکارساز پر می کند.

قوانین تشخیص

قواعد امنیتی یا قوانین تشخیص تهدید، منطق شرطی اعمال شده بر داده های ثبت شده را تعریف می کند که به سیستم اجازه می دهد رویداد مورد علاقه را شناسایی کند. Security Analytics از [قوانین سیگما](#) از پیش بسته بندی شده و منبع باز به عنوان نقطه شروع برای توصیف رویدادهای گزارش مربوطه استفاده می کند. اما قوانین سیگما با فرمت ذاتا منعطف و قابلیت حمل آسان، گزینه هایی را برای وارد کردن و سفارشی کردن قوانین در اختیار کاربران آنالیز امنیتی قرار می دهند. می توانید با استفاده از داشبوردهای OpenSearch یا API از این گزینه ها استفاده کنید.

یافته ها

هر بار که آشکارساز یک قانون را با یک رویداد گزارش مطابقت می دهد، یافته ها ایجاد می شود. یافته ها لزوماً به تهدیدات قریب الوقوع در سیستم اشاره نمی کنند، اما همیشه یک رویداد مورد علاقه را جدا می کنند. از آنجایی که آنها نتیجه یک تعریف خاص برای آشکارساز را نشان می دهند، یافته ها شامل ترکیبی منحصر به فرد از قوانین انتخابی، نوع گزارش و شدت قانون است. به این ترتیب، می توانید یافته های خاصی را در پنجره Findings جستجو کنید، و می توانید یافته های موجود در فهرست را بر اساس شدت و نوع گزارش فیلتر کنید.

هشدارها

هنگام تعریف آشکارساز، می توانید شرایط خاصی را مشخص کنید که باعث ایجاد هشدار می شود. هنگامی که یک رویداد هشدار را راه اندازی می کند، سیستم یک اعلان به یک کانال ترجیحی مانند Slack, Amazon Chime یا ایمیل ارسال می کند. هنگامی که آشکارساز با یک یا چند قانون مطابقت داشته باشد، هشدار می تواند فعال شود. شرایط بیشتر را می توان با شدت قوانین و برچسب ها تنظیم کرد. شما همچنین می توانید یک پیام اعلان با یک موضوع سفارشی و متن پیام ایجاد کنید.

موتور همبستگی

موتور همبستگی به آنالیز امنیتی امکان مقایسه یافته ها از انواع گزارش های مختلف و ترسیم همبستگی بین آنها را می دهد. این امر درک روابط بین یافته های سیستم های مختلف در یک زیرساخت را تسهیل می کند و اطمینان به معنادار بودن یک رویداد و نیاز به توجه را افزایش می دهد.

موتور همبستگی از قوانین همبستگی برای تعریف سناریوهای تهدید شامل انواع مختلف گزارش استفاده می کند. سپس می تواند پرس و جو هایی را بر روی گزارش ها انجام دهد تا یافته های مربوطه را از آن منابع گزارش مختلف مطابقت دهد. برای به تصویر کشیدن روابط بین رویدادهایی که در گزارش های مختلف رخ می دهند، یک نمودار همبستگی نمایشی بصری از یافته ها، ارتباطات آنها و نزدیکی آن اتصالات را ارائه می دهد. در حالی که قوانین همبستگی تعیین می کند که

چه سناریوهای تهدیدی باید جستجو شود، نمودار تصویری را ارائه می دهد که به شما کمک می کند روابط بین یافته های مختلف را در زنجیره ای از رویدادهای امنیتی شناسایی کنید.

نگاشت ها و انواع فیلدها

شما می توانید نحوه ذخیره و نمایه سازی اسناد و فیلدهای آنها را با ایجاد نقشه تعیین کنید. نقشه برداری فهرستی از فیلدها را برای یک سند مشخص می کند. هر فیلد در سند دارای یک نوع فیلد است که نوع داده های فیلد را مشخص می کند. برای مثال، ممکن است بخواهید مشخص کنید که فیلد سال باید از نوع تاریخ (date) باشد. برای کسب اطلاعات بیشتر، [انواع فیلدهای پشتیبانی](#) شده را ببینید.

اگر تازه شروع به ایجاد خوشه و داده خود کرده اید، ممکن است دقیقاً ندانید که داده های شما چگونه باید ذخیره شوند. در این موارد، می توانید از نگاشت های پویا استفاده کنید که به `OpenSearch` می گوید به صورت پویا داده ها و فیلدهای آن را اضافه کند. با این حال، اگر دقیقاً می دانید که داده های شما تحت چه انواعی قرار می گیرند و می خواهید آن استاندارد را اجرا کنید، می توانید از نگاشت های صریح استفاده کنید.

به عنوان مثال، اگر می خواهید نشان دهید که سال باید به جای عدد صحیح (integer) از نوع متن (text) باشد و سن باید یک عدد صحیح (integer) باشد، می توانید این کار را با نگاشت های صریح انجام دهید. با استفاده از نقشه برداری پویا، `OpenSearch` ممکن است سال و سن را به عنوان اعداد صحیح تفسیر کند.

این بخش یک مثال برای نحوه ایجاد یک نگاشت شاخص و نحوه اضافه کردن سندی به آن ارائه می دهد که اعتبار `ip_range` را دریافت می کند.

نقشه برداری پویا

وقتی یک سند را فهرست بندی می کنید، **OpenSearch** فیلدها را به صورت خودکار با نگاشت پویا اضافه می کند. شما همچنین می توانید به صراحت فیلدهایی را به نگاشت فهرست اضافه کنید.

انواع نقشه برداری دینامیک :

Type	Description
null	A <code>null</code> field can't be indexed or searched. When a field is set to null, OpenSearch behaves as if that field has no values.
boolean	OpenSearch accepts <code>true</code> and <code>false</code> as boolean values. An empty string is equal to <code>false</code> .
float	A single-precision 32-bit floating point number.
double	A double-precision 64-bit floating point number.
integer	A signed 32-bit number.
object	Objects are standard JSON objects, which can have fields and mappings of their own. For example, a <code>movies</code> object can have additional properties such as <code>title</code> , <code>year</code> , and <code>director</code> .
array	Arrays in OpenSearch can only store values of one type, such as an array of just integers or strings. Empty arrays are treated as though they are fields with no values.
text	A string sequence of characters that represent full-text values.
keyword	A string sequence of structured characters, such as an email address or ZIP code.
date detection string	Enabled by default, if new string fields match a date's format, then the string is processed as a <code>date</code> field. For example, <code>date: "2012/03/11"</code> is processed as a date.
numeric detection string	If disabled, OpenSearch may automatically process numeric values as strings when they should be processed as numbers. When enabled, OpenSearch can process strings into <code>long</code> , <code>integer</code> , <code>short</code> , <code>byte</code> , <code>double</code> , <code>float</code> , <code>half_float</code> , <code>scaled_float</code> , and <code>unsigned_long</code> . Default is disabled.

نقشه برداری صریح

اگر دقیقاً می‌دانید که انواع داده‌های فیلد شما باید چه باشند، می‌توانید هنگام ایجاد فهرست خود، آنها را در بدنه درخواست خود مشخص کنید.

```
PUT sample-index1
{
  "mappings": {
    "properties": {
      "year": { "type": "text" },
      "age": { "type": "integer" },
      "director": { "type": "text" }
    }
  }
}
```

واکنش

```
{
  "acknowledged": true,
  "shards_acknowledged": true,
  "index": "sample-index1"
}
```

برای افزودن نقشه‌ها به یک فهرست یا جریان داده موجود، می‌توانید با استفاده از روش PUT یا POST HTTP یک درخواست به نقطه پایانی `_mapping` ارسال کنید:

```
POST sample-index1/_mapping
{
  "properties": {
    "year": { "type": "text" },
    "age": { "type": "integer" },
    "director": { "type": "text" }
  }
}
```


تحلیل متن

هنگامی که اسناد را با استفاده از جستجوی متن کامل جستجو می کنید، می خواهید همه نتایج مرتبط را دریافت کنید. اگر به دنبال "راه رفتن" هستید، به نتایجی علاقه مند هستید که حاوی هر شکلی از کلمه هستند، مانند "راه رفتن"، "راه رفتن" یا "راه رفتن". برای تسهیل جستجوی متن کامل، OpenSearch از تجزیه و تحلیل متن استفاده می کند.

هدف تجزیه و تحلیل متن، تقسیم محتوای متن آزاد بدون ساختار سند منبع به دنباله ای از اصطلاحات است که سپس در یک نمایه معکوس ذخیره می شوند. متعاقباً، هنگامی که یک تحلیل متن مشابه برای درخواست کاربر اعمال می شود، توالی عبارات حاصل تطبیق اسناد منبع مرتبط را تسهیل می کند.

از نقطه نظر فنی، فرآیند تحلیل متن شامل چندین مرحله است که برخی از آنها اختیاری هستند:

- قبل از اینکه محتوای متن آزاد را بتوان به کلمات جداگانه تقسیم کرد، ممکن است متن را در سطح کاراکتر اصلاح کنید. هدف اصلی این مرحله اختیاری کمک به توکنایزر (مرحله بعدی در فرآیند تحلیل) است که توکن های بهتری تولید کند. این می تواند شامل حذف برچسب های نشانه گذاری (مانند HTML) یا مدیریت الگوهای کاراکترهای خاص (مانند جایگزینی ایموجی 😊 با متن :slightly_smiling_face:) باشد.
- مرحله بعدی این است که متن آزاد را به کلمات جداگانه تقسیم کنید - نشانه ها. این کار توسط توکنایزر انجام می شود. به عنوان مثال، پس از نشانه گذاری، جمله Actions speak louder than words به نشانه های Actions، speak، louder than، و words تقسیم می شود.
- آخرین مرحله پردازش توکن های فردی با اعمال یک سری فیلتر توکن است. هدف تبدیل هر توکن به شکلی قابل پیش بینی است که مستقیماً در ایندکس ذخیره می شود، به عنوان مثال، با تبدیل آنها به حروف کوچک یا اجرای stemming (کاهش کلمه به ریشه آن). به عنوان مثال، نشانه Actions تبدیل به action، louder تبدیل به loud و words تبدیل به word می شود.

آنالیزرها

در OpenSearch، انتزاعی که تجزیه و تحلیل متن را در بر می گیرد، تحلیلگر نامیده می شود. هر تحلیلگر شامل اجزای زیر است که به صورت متوالی اعمال می شوند:

- **فیلترهای کاراکتر:** ابتدا یک فیلتر کاراکتر متن اصلی را به صورت جریانی از کاراکترها دریافت می کند و کاراکترهایی را در متن اضافه، حذف یا تغییر می دهد. برای مثال، یک فیلتر کاراکتر می تواند کاراکترهای HTML را از یک رشته حذف کند تا متن `<p>Actions speak louder than words</p>` صحبت کند، به `n\عملکردها بلندتر از کلمات صحبت می کنند. n\` خروجی یک فیلتر کاراکتر جریانی از کاراکترها است.

- **Tokenizer:** در مرحله بعد، یک توکنایزر جریانی از کاراکترها را دریافت می کند که توسط فیلتر کاراکتر پردازش شده است و متن را به نشانه های فردی (معمولاً کلمات) تقسیم می کند. برای مثال، یک نشانه ساز می تواند متن را در فضای سفید تقسیم کند تا متن قبلی به [Actions, speak, louder, than, words] تبدیل شود. Tokenizers همچنین ابرداده هایی را در مورد نشانه ها مانند موقعیت های شروع و پایان آنها در متن حفظ می کنند. خروجی توکنایزر یک جریان توکن است.

- **فیلترهای توکن:** در آخر، یک فیلتر توکن جریان توکن ها را از توکنایزر دریافت می کند و توکن ها را اضافه، حذف یا تغییر می دهد. به عنوان مثال، یک فیلتر نشانه ممکن است نشانه ها را کوچک کند تا Actions تبدیل به action شود، کلمات توفقی مانند than را حذف کند، یا مترادف هایی مانند talk برای کلمه speak اضافه کند.

آنالایزهای داخلی

جدول زیر تحلیلگرهای داخلی را که OpenSearch ارائه می دهد فهرست می کند. آخرین ستون جدول حاوی نتیجه اعمال تحلیلگر در رشته It's fun to contribute a brand-new PR or 2 to OpenSearch! است.

Analyzer	Analysis performed	Analyzer output
Standard (default)	<ul style="list-style-type: none"> - Parses strings into tokens at word boundaries - Removes most punctuation - Converts tokens to lowercase 	[it's, fun, to, contribute, a, brand, new, pr, or, 2, to, opensearch]
Simple	<ul style="list-style-type: none"> - Parses strings into tokens on any non-letter character - Removes non-letter characters - Converts tokens to lowercase 	[it, s, fun, to, contribute, a, brand, new, pr, or, to, opensearch]
Whitespace	<ul style="list-style-type: none"> - Parses strings into tokens on white space 	[It's, fun, to, contribute, a, brand-new, PR, or, 2, to, OpenSearch!]
Stop	<ul style="list-style-type: none"> - Parses strings into tokens on any non-letter character - Removes non-letter characters - Removes stop words - Converts tokens to lowercase 	[s, fun, contribute, brand, new, pr, opensearch]
Keyword (no-op)	<ul style="list-style-type: none"> - Outputs the entire string unchanged 	[It's fun to contribute a brand-new PR or 2 to OpenSearch!]
Pattern	<ul style="list-style-type: none"> - Parses strings into tokens using regular expressions - Supports converting strings to lowercase - Supports removing stop words 	[it, s, fun, to, contribute, a, brand, new, pr, or, 2, to, opensearch]
Language	Performs analysis specific to a certain language (for example, english).	[fun, contribut, brand, new, pr, 2, opensearch]
Fingerprint	<ul style="list-style-type: none"> - Parses strings on any non-letter character - Normalizes characters by converting them to ASCII - Converts tokens to lowercase - Sorts, deduplicates, and concatenates tokens into a single token - Supports removing stop words 	[2 a brand contribute fun it's new opensearch or pr to] Note that the apostrophe was converted to its ASCII counterpart.

تست یک آنالایزر

برای آزمایش یک آنالایزر داخلی و مشاهده لیست نشانه هایی که هنگام نمایه شدن یک سند تولید می کند، می توانید از Analyze API استفاده کنید.

تحلیلگر و متن مورد تجزیه و تحلیل را در درخواست مشخص کنید:

```
GET /_analyze
{
  "analyzer" : "standard",
  "text" : "Let's contribute to OpenSearch!"
}
```

تصویر زیر رشته پرس و جو را نشان می دهد.

L	e	t	'	s		c	o	n	t	r	i	b	u	t	e		t	o		O	p	e	n	S	e	a	r	c	h	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

پاسخ شامل هر نشانه و آفست های شروع و پایان آن است که با شاخص شروع در رشته اصلی (شامل) و شاخص پایانی (انحصاری) مطابقت دارد:

```
{
  "tokens": [
    {
      "token": "let's",
      "start_offset": 0,
      "end_offset": 5,
      "type": "<ALPHANUM>",
      "position": 0
    },
    {
      "token": "contribute",
      "start_offset": 6,
      "end_offset": 16,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "to",
      "start_offset": 17,
      "end_offset": 19,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "opensearch",
      "start_offset": 20,
      "end_offset": 30,
      "type": "<ALPHANUM>",
      "position": 3
    }
  ]
}
```

پرس و جو از DSL

OpenSearch یک زبان جستجو به نام query-domain-specific language (DSL) ارائه می‌کند که می‌توانید برای جستجوی داده‌های خود از آن استفاده کنید. Query DSL یک زبان منعطف با رابط JSON است.

با Query DSL، باید یک پرس و جو را در پارامتر query جستجو مشخص کنید. یکی از ساده‌ترین جستجوها در OpenSearch از عبارت match_all استفاده می‌کند که با تمام اسناد موجود در یک فهرست مطابقت دارد:

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

یک پرس و جو می‌تواند از جمله‌های پرس و جو زیادی تشکیل شده باشد. شما می‌توانید جملات پرس و جو را برای ایجاد پرس و جوهای پیچیده ترکیب کنید.

به طور کلی، می‌توانید پرس و جوها را به دو دسته دسته بندی کنید - پرس و جوهای برگ و پرس و جوهای مرکب:

پرس و جوهای برگ:

پرس و جوهای برگ برای یک مقدار مشخص در یک فیلد یا فیلدهای خاص جستجو می‌کنند. شما می‌توانید از پرس و جوهای برگ به تنهایی استفاده کنید. آنها شامل انواع پرس و جو زیر هستند:

- **پرس و جوهای متن کامل:** از پرس و جوهای تمام متن برای جستجوی اسناد متنی استفاده کنید. برای جستجوی فیلد متنی تجزیه و تحلیل شده، پرس و جوهای متن کامل رشته پرس و جو را با استفاده از همان تحلیلگری که هنگام نمایه سازی فیلد استفاده می‌شد، به عبارات تقسیم می‌کنند. برای جستجوی مقدار دقیق، پرس و جوهای متن کامل بدون اعمال تجزیه و تحلیل متن به دنبال مقدار مشخص شده می‌گردند.
- **پرس و جوهای سطح عبارت:** از پرس و جوهای سطح اصطلاح برای جستجوی اسناد برای یک عبارت دقیق، مانند شناسه یا محدوده مقدار استفاده کنید. پرس و جوهای سطح اصطلاح، عبارات جستجو را تجزیه و تحلیل نمی‌کنند یا نتایج را بر اساس امتیاز مربوطه مرتب نمی‌کنند.
- **جستارهای جغرافیایی و xy:** از پرس و جوهای جغرافیایی برای جستجوی اسنادی که شامل داده های جغرافیایی هستند استفاده کنید. از پرس و جوهای xy برای جستجوی اسنادی که شامل نقاط و اشکال در یک سیستم مختصات دو بعدی هستند، استفاده کنید.

- **پیوستن به پرس و جوها:** برای جستجوی فیلدهای تودرتو یا برگرداندن اسناد والد و فرزندی که با یک جستار خاص مطابقت دارند از جستارهای الحاقی استفاده کنید. انواع پرس و جوهایی پیوستن عبارتند از پرس و جوهایی تودرتو (nested)، has_child، has_parent و parent_id.
 - **Span query:** از span query برای انجام جستجوهای موقعیتی دقیق استفاده کنید. پرس و جوهایی Span عبارتند از پرس و جوهایی سطح پایین و خاص که کنترلی بر ترتیب و نزدیکی عبارت‌های پرس و جو مشخص می‌کنند. آنها در درجه اول برای جستجوی اسناد قانونی استفاده می‌شوند.
 - **پرس و جوهایی تخصصی:** پرس و جوهایی تخصصی شامل تمام انواع دیگر پرس و جو می‌شود (distance_feature، more_like_this، percolate، rank_feature، script، script_score و wrapper).
- پرس و جوهایی مرکب: پرس و جوهایی مرکب به عنوان لفاف بندی برای بندهای چند برگ یا ترکیبی عمل می‌کنند تا نتایج خود را ترکیب کنند یا رفتار خود را تغییر دهند. آنها شامل Boolean، حداکثر تفکیک، امتیاز ثابت، امتیاز تابع و انواع پرس و جو تقویت کننده هستند.

تجمعات (Aggregations)

OpenSearch فقط برای جستجو نیست. تجمیع‌ها به شما امکان می‌دهند از موتور قدرتمند تجزیه و تحلیل OpenSearch برای تجزیه و تحلیل داده‌های خود و استخراج آمار از آن استفاده کنید.

موارد استفاده از تجمیع‌ها از تجزیه و تحلیل داده‌ها در زمان واقعی تا انجام برخی اقدامات تا استفاده از داشبوردهای جستجوی باز برای ایجاد داشبورد تجسم متفاوت است.

OpenSearch می‌تواند جمع‌آوری‌ها را در مجموعه‌های داده عظیم در میلی ثانیه انجام دهد. در مقایسه با پرس و جوها، تجمیع چرخه های CPU و حافظه بیشتری مصرف می‌کنند.

تجمیع در فیلدهای متنی

به طور پیش فرض، OpenSearch از تجمیع‌ها در یک فیلد متنی پشتیبانی نمی‌کند. از آنجایی که فیلدهای متنی نشانه گذاری می‌شوند، یک ادغام در یک فیلد متنی باید فرآیند توکن سازی را به رشته اصلی خود برگرداند و سپس یک تجمیع را بر اساس آن فرموله کند. این نوع عملیات حافظه قابل توجهی را مصرف می‌کند و عملکرد خوشه را کاهش می‌دهد.

در حالی که می‌توانید با تنظیم پارامتر fielddata روی true در نگاشت، تجمیع‌ها را در فیلدهای متنی فعال کنید، تجمیع‌ها همچنان بر اساس کلمات نشانه‌گذاری شده هستند و نه بر اساس متن خام.

توصیه می‌کنیم یک نسخه خام از فیلد متنی را به عنوان یک فیلد کلمه کلیدی نگه دارید که می‌توانید آن را جمع‌آوری کنید.

در این حالت، می‌توانید به‌جای اینکه در فیلد عنوان (title)، تجمیع‌ها را در قسمت title.raw انجام دهید:

```
PUT movies
{
  "mappings": {
    "properties": {
      "title": {
        "type": "text",
        "fielddata": true,
        "fields": {
          "raw": {
            "type": "keyword"
          }
        }
      }
    }
  }
}
```

ساختار تجمع عمومی

ساختار یک پرس و جو تجمع به شرح زیر است:

```
GET _search
{
  "size": 0,
  "aggs": {
    "NAME": {
      "AGG_TYPE": {}
    }
  }
}
```

اگر فقط به نتیجه تجمیع و نه به نتایج پرس و جو علاقه دارید، اندازه را روی ۰ تنظیم کنید.

در ویژگی **aggs** (در صورت تمایل می‌توانید از تجمیع استفاده کنید) می‌توانید هر تعداد تجمع را تعریف کنید. هر تجمیع با نام خود و یکی از انواع تجمیع‌هایی که **OpenSearch** پشتیبانی می‌کند تعریف می‌شود.

نام تجمیع به شما کمک می‌کند تا بین تجمیع‌های مختلف در پاسخ تمایز قائل شوید. ویژگی **AGG_TYPE** جایی است که نوع تجمع را مشخص می‌کنید.

مثالی از تجمع

این بخش از داده‌های تجارت الکترونیک نمونه و داده‌های گزارش وب داشبوردهای OpenSearch استفاده می‌کند. برای افزودن داده‌های نمونه، وارد داشبوردهای OpenSearch شوید، صفحه اصلی را انتخاب کنید و سپس داده‌های نمونه را امتحان کنید. برای نمونه سفارش‌های تجارت الکترونیک و نمونه گزارش‌های وب، افزودن داده را انتخاب کنید.

میانگین

برای یافتن مقدار متوسط فیلد `taxful_total_price`:

```
GET opensearch_dashboards_sample_data_ecommerce/_search
{
  "size": 0,
  "aggs": {
    "avg_taxful_total_price": {
      "avg": {
        "field": "taxful_total_price"
      }
    }
  }
}
```

پاسخ مثال :

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 4675,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "avg_taxful_total_price" : {
      "value" : 75.05542864304813
    }
  }
}
```

بلوک تجمیع در پاسخ، مقدار متوسط را برای قسمت `taxful_total_price` نشان می‌دهد.

داکیومنت:

سند واحدی است که اطلاعات (متن یا داده های ساخت یافته) را ذخیره می کند. در **OpenSearch**، اسناد در قالب **JSON** ذخیره می شوند.

ما می توانیم یک سند را به چند روش در نظر بگیریم:

در پایگاه داده دانشجویان، یک سند ممکن است نشان دهنده یک دانش آموز باشد.

وقتی اطلاعاتی را جستجو می کنید، **OpenSearch** اسناد مربوط به جستجوی شما را برمی گرداند. یک سند نشان دهنده یک ردیف در یک پایگاه داده سنتی است. به عنوان مثال، در یک پایگاه داده مدرسه، یک سند ممکن است یک دانش آموز را نشان دهد و حاوی داده های زیر باشد.

ID	Name	GPA	Graduation year
1	John Doe	3.89	2022

در اینجا شکل این سند در قالب **JSON** است:

```
{
  "name": "John Doe",
  "gpa": 3.89,
  "grad_year": 2022
}
```

اینکدس:

نمایه مجموعه ای از اسناد است.

در پایگاه داده دانشجویان، یک نمایه نشان دهنده همه دانش آموزان در پایگاه داده است. وقتی اطلاعاتی را جستجو می کنید، داده های موجود در یک فهرست را پرس و جو می کنید. ایندکس یک جدول پایگاه داده را در یک پایگاه داده سنتی نشان می دهد. به عنوان مثال، در یک پایگاه داده مدرسه، یک نمایه ممکن است شامل همه دانش آموزان مدرسه باشد.

ID	Name	GPA	Graduation year
1	John Doe	3.89	2022
2	Jonathan Powers	3.85	2025
3	Jane Doe	3.52	2024

جدول معکوس:

یک نمایه OpenSearch از ساختار داده ای به نام شاخص معکوس استفاده می کند. یک نمایه معکوس کلمات را به اسنادی که در آنها وجود دارند نگاشت می کند. به عنوان مثال، یک شاخص شامل دو سند زیر را در نظر بگیرید:

- Document 1: "Beauty is in the eye of the beholder"
- Document 2: "Beauty and the beast"

یک نمایه معکوس برای چنین نمایه ای، کلمات را به اسنادی که در آنها قرار دارند نگاشت می کند:

Word	Document
beauty	1, 2
is	1
in	1
the	1, 2
eye	1
of	1
the	1
beholder	1
and	2
beast	2

علاوه بر شناسه سند، OpenSearch موقعیت کلمه را در سند برای اجرای عبارت عبارت ذخیره می کند، جایی که کلمات باید در کنار یکدیگر ظاهر شوند.

ارتباط:

هنگامی که یک سند را جستجو می کنید، OpenSearch کلمات موجود در پرس و جو را با کلمات موجود در اسناد مطابقت می دهد. به عنوان مثال، اگر فهرست توصیف شده در بخش قبل را برای کلمه beauty جستجو کنید، OpenSearch اسناد ۱ و ۲ را برمی گرداند. به هر سند یک امتیاز مرتبط اختصاص داده می شود که به شما می گوید چقدر سند با درخواست مطابقت دارد.

کلمات منفرد در یک عبارت جستجو را اصطلاحات جستجو می گویند. هر عبارت جستجو بر اساس قوانین زیر امتیازدهی می شود:

(۱) عبارت جستجویی که بیشتر در یک سند وجود دارد، امتیاز بیشتری کسب می کند. سندی در مورد سگ ها که بارها از کلمه سگ استفاده می کند، احتمالاً مرتبط تر از سندی است که تعداد کمتری کلمه سگ را در خود دارد. این عبارت جزء فرکانس امتیاز است.


(۲) عبارت جستجویی که در اسناد بیشتری وجود دارد، امتیاز کمتری دریافت می کند. پرس و جو برای عبارات آبی و axolotl باید اسنادی را که حاوی axolotl هستند بر کلمه احتمالاً رایج تر آبی ترجیح دهد. این جزء معکوس فرکانس سند نمره است.

(۳) یک تطابق در یک سند طولانی تر باید امتیاز کمتری نسبت به یک سند کوتاه تر داشته باشد. سندی که حاوی یک فرهنگ لغت کامل باشد، با هر کلمه ای مطابقت دارد، اما به کلمه خاصی مرتبط نیست. این با مولفه نرمال سازی طول امتیاز مطابقت دارد.

این سیستم از الگوریتم رتبه بندی BM25 برای محاسبه امتیازات مربوط به اسناد استفاده می کند و سپس نتایج مرتب شده بر اساس ارتباط را برمی گرداند.

بخش عملی ارتباط با open search


برای اتصال به این سیستم، باید از داکر استفاده نمود. ما به یک فایل خاص به نام فایل Compose نیاز داریم که Docker Compose از آن برای تعریف و ایجاد کانتینرها در کلاستر ما استفاده می‌کند. پروژه OpenSearch یک نمونه فایل Compose را ارائه می‌دهد که می‌توانیم برای شروع از آن استفاده کنیم.

 docker-compose.yml

4/6/2024 11:55 PM

Yaml Source File

4 KB

 docker desktop

Search for images, containers, volumes, extensions... **Ctrl+K**

Sign in

Containers

Give feedback





Containers CPU usage ⓘ
4.55% / 1200% (12 CPUs available)

Containers memory usage ⓘ
2.58GB / 7.5GB

Show charts

Search

Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 opensearch		Running (3/3)	4.55%		36 minutes ago	■ ⓘ ⌵
<input type="checkbox"/>	 node1 6964bde4a097 ⓘ	opensearchproject/opensearch:latest	Running	2.42%	9200:9200 ⓘ Show all ports (2)	36 minutes ago	■ ⓘ ⌵
<input type="checkbox"/>	 node2 c2b4d06d6edc ⓘ	opensearchproject/opensearch:latest	Running	1.94%		36 minutes ago	■ ⓘ ⌵
<input type="checkbox"/>	 dashboards e1aed6e25ede ⓘ	opensearchproject/opensearch-dashboi	Running	0.19%	5601:5601 ⓘ	36 minutes ago	■ ⓘ ⌵

Showing 4 items

با وارد کردن دستور زیر سرویس‌ها فعال و اجرا می‌شوند.

```
PS D:\OpenSearch> docker-compose up
[+] Running 3/0
✔ Container opensearch-node1      Running
✔ Container opensearch-dashboards Running
✔ Container opensearch-node2      Running
Attaching to opensearch-dashboards, opensearch-node1, opensearch-node2
```

برای اینکه مطمئن شویم سرویس‌ها به درستی اجرا شده است دستور زیر را در مرورگر وارد می‌کنیم:

دقت شود پورت ۹۲۰۰ و ۹۶۰۰ برای نود ۱ و ۲ و پورت ۵۶۰۱ برای داشبورد می‌باشد.

Localhost:9200

```
.
{
  "name" : "opensearch-node1",
  "cluster_name" : "opensearch-cluster",
  "cluster_uuid" : "HejHXqdtRWK0fYVidt-e2Q",
  "version" : {
    "distribution" : "opensearch",
    "number" : "2.13.0",
    "build_type" : "tar",
    "build_hash" : "7ec678d1b7c87d6e779fdef94e33623e1f1e2647",
    "build_date" : "2024-03-26T00:02:39.659767978Z",
    "build_snapshot" : false,
    "lucene_version" : "9.10.0",
    "minimum_wire_compatibility_version" : "7.10.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

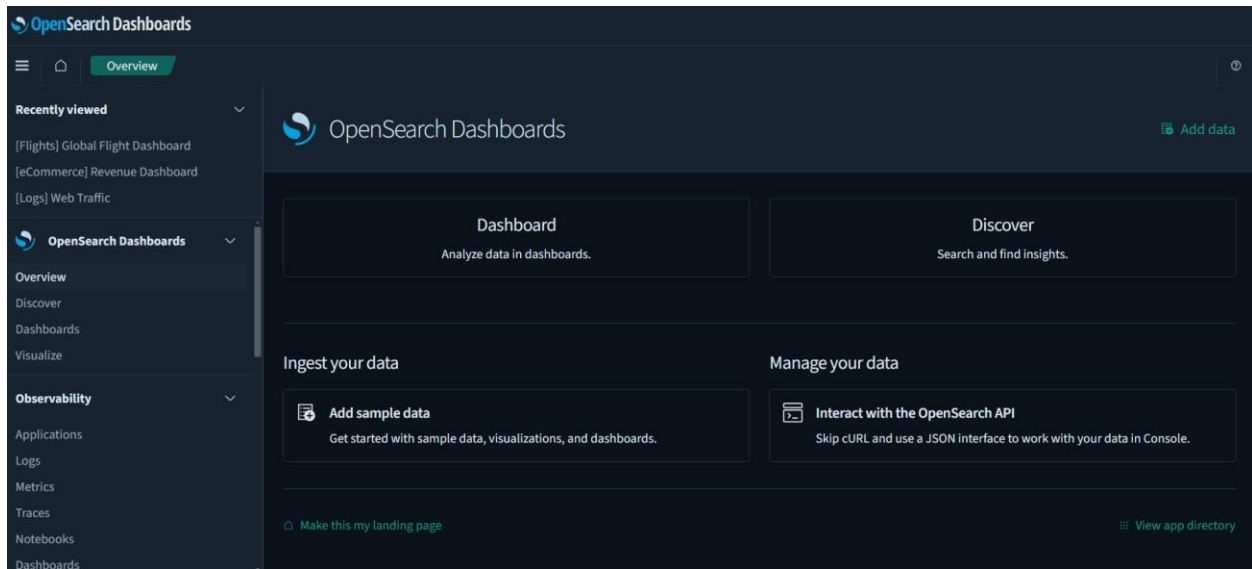
و یا با وارد کردن دستور زیر در ترمینال:

docker-compose up -d

```
$ docker-compose ps
NAME                COMMAND                                SERVICE            STATUS            PORTS
opensearch-dashboards  ./opensearch-dashbo...               opensearch-dashboards  running           0.0.0.0:5601->5601
opensearch-node1      ./opensearch-docker...               opensearch-node1      running           0.0.0.0:9200->9200
opensearch-node2      ./opensearch-docker...               opensearch-node2      running           9200/tcp, 9300/tcp
```

با وارد کردن دستور زیر به داشبورد سیستم متصل می‌شویم:

localhost:5601



زبان‌های مورد پشتیبانی برای اتصال Open search :

High-level Python client

Low-level Python client

Opensearch-py-ml

Java high-level REST client

Java client

JavaScript client

Go client

Ruby client

PHP client

.NET clients

Rust client

در این پروژه تمامی کدنویسی‌ها با استفاده از زبان پایتون (سطح بالا) می‌باشد:

```
pip install opensearch-dsl
```

بعد از نصب می‌توانیم از کتابخانه‌ها برای پروژه استفاده کنیم:

```
from opensearchpy import OpenSearch
```

```
from opensearch_dsl import Search
```

برای اتصال به میزبان **OpenSearch** پیش فرض، اگر از افزونه **Security** استفاده می‌شود، یک شی کلاینت با **SSL** فعال ایجاد کنید.

```
host = 'localhost'
```

```
port = 9200
```

```
auth = ('admin', 'admin') # For testing only. Don't store credentials in code.
```

```
ca_certs_path = '/full/path/to/root-ca.pem' # Provide a CA bundle if you use intermediate CAs  
with your root CA.
```

```
# Create the client with SSL/TLS enabled, but hostname verification disabled.
```

```
client = OpenSearch(
```

```
    hosts = [{'host': host, 'port': port}],
```

```
    http_compress = True, # enables gzip compression for request bodies
```

```
    http_auth = auth,
```

```
    use_ssl = True,
```

```
    verify_certs = True,
```

```
    ssl_assert_hostname = False,
```

```
    ssl_show_warn = False,
```

```
    ca_certs = ca_certs_path
```

```
)
```

اگر گواهینامه مشتری خود را دارید، آنها را در پارامترهای `client_key_path` و `client_cert_path` مشخص کنید:

```
host = 'localhost'

port = 9200

auth = ('admin', 'admin') # For testing only. Don't store credentials in code.

ca_certs_path = '/full/path/to/root-ca.pem' # Provide a CA bundle if you use intermediate CAs
with your root CA.

# Optional client certificates if you don't want to use HTTP basic authentication.
client_cert_path = '/full/path/to/client.pem'
client_key_path = '/full/path/to/client-key.pem'

# Create the client with SSL/TLS enabled, but hostname verification disabled.
client = OpenSearch(
    hosts = [{'host': host, 'port': port}],
    http_compress = True, # enables gzip compression for request bodies
    http_auth = auth,
    client_cert = client_cert_path,
    client_key = client_key_path,
    use_ssl = True,
    verify_certs = True,
    ssl_assert_hostname = False,
    ssl_show_warn = False,
    ca_certs = ca_certs_path
)
```

اگر از افزونه **Security** استفاده نمی کنید، یک شی کلاینت با **SSL** غیرفعال ایجاد کنید:

نکته: همچنین می توانیم در ابتدای کانفیگ قسمت **Security** را غیرفعال کنیم.

```
host = 'localhost'
```

```
port = 9200
```

```
# Create the client with SSL/TLS and hostname verification disabled.
```

```
client = OpenSearch(
```

```
    hosts = [{'host': host, 'port': port}],
```

```
    http_compress = True, # enables gzip compression for request bodies
```

```
    use_ssl = False,
```

```
    verify_certs = False,
```

```
    ssl_assert_hostname = False,
```

```
    ssl_show_warn = False
```

```
)
```

ایجاد شاخص

برای ایجاد یک فهرست **OpenSearch**، از متد `client.indices.create()` استفاده کنید. می توانید از کد زیر برای ساخت یک شی **JSON** با تنظیمات سفارشی استفاده کنید:

```
index_name = 'test'
```

```
index_body = {
```

```
    'settings': {
```

```
        'index': {
```

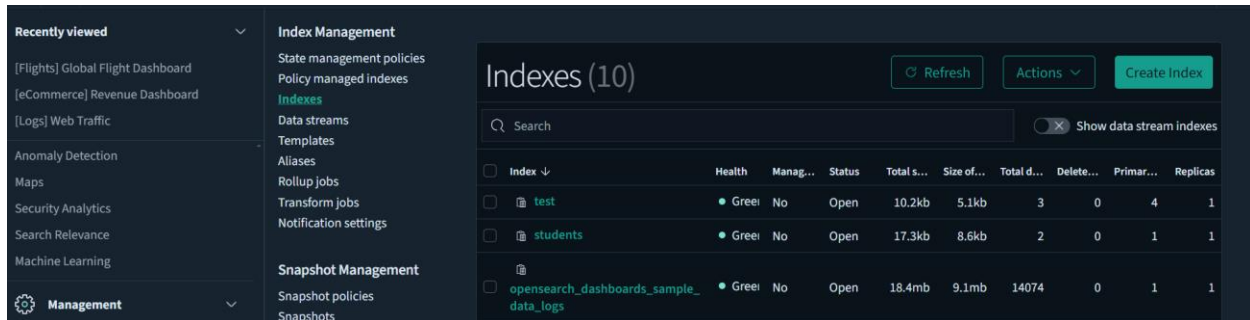
```
            'number_of_shards': 4
```

```
        }
```

```
    }
```

```
}
```

```
response = client.indices.create(index_name, body=index_body)
```

می توانید با گسترش کلاس Document یک کلاس برای نشان دادن اسنادی که در OpenSearch نمایه می کنید ایجاد کنید:

```
class Movie(Document):
```

```
    title = Text(fields={'raw': Keyword()})
```

```
    director = Text()
```

```
    year = Text()
```

```
class Index:
```

```
    name = index_name
```

```
    def save(self, ** kwargs):
```

```
        return super(Movie, self).save(** kwargs)
```

متد **save()**:

برای ایندکس کردن یک سند، یک شی از کلاس جدید ایجاد کنید و متد `save()` آن را فراخوانی کنید:

```
# Set up the opensearch-py version of the document
```

```
Movie.init(using=client)
```

```
doc = Movie(meta={'id': 1}, title='Moneyball', director='Bennett Miller', year='2011')
```

```
response = doc.save(using=client)
```

متد bulk :

با استفاده از متد bulk کلاینت می‌توانید چندین عملیات را همزمان انجام دهید. عملیات ممکن است از یک نوع یا از انواع مختلف باشد. توجه داشته باشید که عملیات باید با `\n` از هم جدا شوند و کل رشته باید یک خط باشد:

```
# Add sample documents
if not client.exists(index=index_name, id=1):
    doc1 = Movie(meta={'id': 1}, title='Moneyball', director='Bennett Miller', year='2011')
    doc1.save(using=client)

movies_bulk = [
    '{ "index" : { "_index" : "test", "_id" : "2" } }',
    '{ "title" : "Interstellar", "director" : "Christopher Nolan", "year" : "2014"}',
    '{ "create" : { "_index" : "test", "_id" : "3" } }',
    '{ "title" : "Star Trek Beyond", "director" : "Justin Lin", "year" : "2015"}',
    '{ "update" : { "_id" : "3", "_index" : "test" } }',
    '{ "doc" : { "year" : "2016" } }'
]
client.bulk('\n'.join(movies_bulk))
```

شما می‌توانید از کلاس Search برای ساخت یک پرس و جو استفاده کنید. کد زیر یک کوئری بولی با فیلتر ایجاد می‌کند:

```
s = Search(using=client, index=index_name) \
    .filter("term", year="2011") \
    .query("match", title="Moneyball")
```

```
response = s.execute()
```

متد client.delete() :

می‌توانید یک سند را با استفاده از متد `client.delete()` حذف کنید:

```
response = client.delete(
    index = 'my-dsl-index',
    id = '1'
)
```

می توانید با استفاده از متد `client.indices.delete()` یک فهرست را حذف کنید:

```
response = client.indices.delete(  
    index = 'my-dsl-index'  
)
```

رابط کاربری:

در این پروژه تصمیم گرفته شد رابط کاربری با استفاده از `streamlit` انجام شود.

```
streamlit run app.py
```

خروجی:

`http://localhost:8501`

با توضیحات بالا یک مثال ساده ببینیم:

Deploy ⋮

Movie Search App

Search for a movie

Interstellar

Search

Search results for: **Interstellar**

Title: Interstellar

Director: Christopher Nolan

Year: 2014

در ادامه قدرت سرچ و بازیابی این سیستم به وضوح نشان داده خواهد شد.

بازیابی اطلاعات:

جستجو کردن:

OpenSearch ویژگی‌های بسیاری را برای سفارشی کردن موارد استفاده از جستجو و بهبود ارتباط جستجو فراهم می‌کند.

روش‌های جستجو

OpenSearch از روش‌های جستجوی زیر پشتیبانی می‌کند:

❖ جستجوی واژگانی سنتی

جستجوی کلمه کلیدی (BM25): مجموعه سند را برای کلماتی که در پرس و جو ظاهر می‌شوند جستجو می‌کند.

❖ جستجوی مبتنی بر یادگیری ماشینی (ML)

جستجوی برداری

جستجوی K-NN: K-نزدیکترین همسایه یک عبارت جستجو را در میان شاخصی از بردارها جستجو می‌کند.

جستجوی عصبی

جستجوی عصبی تولید جاسازی‌های برداری را در زمان مصرف و جستجوی آنها در زمان جستجو را تسهیل می‌کند. جستجوی عصبی به شما امکان می‌دهد مدل‌های ML را در جستجوی خود ادغام کنید و به عنوان چارچوبی برای اجرای سایر روش‌های جستجو عمل می‌کند. روش‌های جستجوی زیر بر روی جستجوی عصبی ساخته شده‌اند:

- جستجوی معنایی: معنای کلمات را در زمینه جستجو در نظر می‌گیرد. از بازیابی متراکم بر اساس مدل‌های جاسازی متن برای جستجوی داده‌های متنی استفاده می‌کند.
- جستجوی چندوجهی: از مدل‌های تعبیه چند وجهی برای جستجوی داده‌های متن و تصویر استفاده می‌کند.
- جستجوی پراکنده عصبی: از بازیابی پراکنده بر اساس مدل‌های جاسازی پراکنده برای جستجوی داده‌های متنی استفاده می‌کند.

- [جستجوی ترکیبی](#): جستجوی سنتی و جستجوی برداری را برای بهبود ارتباط جستجو ترکیب می‌کند.
- [جستجوی مکالمه](#): یک جستجوی مولد تقویت شده با بازیابی را پیاده سازی می‌کند.

فراگیری ماشین (Machine Learning)

[افزونه ML Commons](#) ویژگی‌های یادگیری ماشین (ML) را در OpenSearch ارائه می‌دهد.

یکپارچه سازی مدل های ML:

برای جستجوی مبتنی بر مدل ML، می‌توانید از یک مدل از پیش آموزش دیده ارائه شده توسط OpenSearch استفاده کنید، مدل خود را در خوشه OpenSearch آپلود کنید، یا به یک مدل پایه که در یک پلت فرم خارجی میزبانی شده است متصل شوید. در OpenSearch نسخه ۲.۹ و جدیدتر، می‌توانید مدل‌های محلی و خارجی را به طور همزمان در یک کلاستر ادغام کنید.

مدیریت مدل های ML در داشبوردهای OpenSearch:

مدیران خوشه‌های ML می‌توانند از داشبوردهای OpenSearch برای بررسی و مدیریت وضعیت مدل‌های ML در حال اجرا در یک خوشه استفاده کنند.

پشتیبانی از الگوریتم‌ها:

ML Commons از الگوریتم‌های مختلفی برای کمک به آموزش مدل‌های ML و پیش‌بینی یا آزمایش پیش‌بینی‌های مبتنی بر داده بدون مدل پشتیبانی می‌کند.

ML Commons API:

ML Commons مجموعه‌ای از API های REST خود را ارائه می‌دهد. برای اطلاعات بیشتر به Commons API مراجعه کنید.

زبان‌های جستجوی در OpenSearch :

در OpenSearch، می‌توانید از زبان‌های جستجوی زیر برای جستجوی داده‌های خود استفاده کنید:

- **زبان خاص دامنه پرس و جو (DSL):** زبان پرس و جو اولیه OpenSearch که از ایجاد پرس و جوهای پیچیده و کاملاً قابل تنظیم پشتیبانی می‌کند.
- **Query string query language:** یک زبان پرس و جو کوچک شده که می‌توانید در پارامتر پرس و جو درخواست جستجو یا در داشبوردهای OpenSearch استفاده کنید.
- **SQL:** یک زبان پرس و جو سنتی که شکاف بین مفاهیم پایگاه داده سنتی رابطه‌ای و انعطاف‌پذیری ذخیره‌سازی اسناد مداربسته OpenSearch را پر می‌کند.
- **زبان پردازش لوله ای (PPL):** زبان اصلی مورد استفاده با قابلیت مشاهده در OpenSearch. PPL از یک نحو لوله ای استفاده می‌کند که دستورات را به یک پرس و جو زنجیره می‌دهد.
- **Dashboards Query Language (DQL):** یک زبان پرس و جو مبتنی بر متن ساده برای فیلتر کردن داده‌ها در داشبوردهای OpenSearch.

عملکرد جستجو

OpenSearch چندین راه برای بهبود عملکرد جستجو ارائه می‌دهد:

- **جستجوی ناهمزمان:** جستارهای پرمصرف منابع را به صورت ناهمزمان اجرا می‌کند.
 - **جستجوی بخش همزمان:** بخش‌ها را همزمان جستجو می‌کند.
- OpenSearch از عملیات رایج زیر در نتایج جستجو پشتیبانی می‌کند:

- Paginate
- Sort
- Highlight search terms
- Autocomplete
- Did-you-mean

Searching data

آنچه کاربران از موتورهای جستجو انتظار دارند در طول سال ها تکامل یافته است. فقط بازگرداندن سریع نتایج مرتبط دیگر برای اکثر کاربران کافی نیست. اکنون کاربران به دنبال روش هایی هستند که به آنها امکان می دهد نتایج مرتبط تری را دریافت کنند، نتایج را مرتب و سازماندهی کنند و پرس و جوی خود را برجسته کنند. OpenSearch شامل ویژگی های زیادی است که در جدول زیر توضیح داده شده است که تجربه جستجو را بهبود می بخشد.

Feature	Description
Autocomplete functionality	Suggest phrases as the user types.
Did-you-mean functionality	Check spelling of phrases as the user types.
Paginate results	Rather than a single, long list, separate search results into pages.
Sort results	Allow sorting of results by different criteria.
Highlight query matches	Highlight the search term in the results.

1.1 Paginate results

- The from and size parameters
- The scroll search operation
- The search_after parameter
- Point in Time with search_after

• The from and size parameters

پارامتر from شماره سندی است که می خواهید نتایج را از آنجا شروع کنید. پارامتر اندازه تعداد نتایجی است که می خواهید نشان دهید. آنها با هم به شما اجازه می دهند زیرمجموعه ای از نتایج جستجو را برگردانید. به عنوان مثال، اگر مقدار اندازه ۱۰ و مقدار from 0 باشد، ۱۰ نتیجه اول را مشاهده می کنید. اگر مقدار از را به ۱۰ تغییر دهید، ۱۰ نتیجه بعدی را مشاهده خواهید کرد (زیرا نتایج دارای شاخص صفر هستند). بنابراین اگر می خواهید نتایجی را ببینید که از نتیجه ۱۱ شروع می شود، از باید ۱۰ باشد.

Movie Search App

Search for a movie by title

Interstellar

Enter 'from' index (starting index)

0

Enter 'size' (number of results per page)

1

Search

Search results for: **Interstellar** (From index 0, Size 1)

Title: Interstellar

Director: Christopher Nolan

Year: 2014

Total Pages: 2

1

2

Scroll search •

پارامترهای **from** و **size** به شما این امکان را می دهند که نتایج جستجوی خود را صفحه بندی کنید اما با محدودیت ۱۰۰۰۰ نتیجه در یک زمان. اگر نیاز دارید حجم داده های بزرگتر از ۱ PB را از مثلاً یک کار یادگیری ماشینی درخواست کنید، به جای آن از عملیات اسکرول استفاده کنید. عملیات اسکرول به شما امکان می دهد تا تعداد نامحدودی از نتایج را درخواست کنید. برای استفاده از عملیات اسکرول، یک پارامتر اسکرول را به سرفصل درخواست اضافه کنید با یک زمینه جستجو که به **OpenSearch** می گوید چه مدت باید به پیمایش ادامه دهید. این زمینه جستجو باید به اندازه کافی طولانی باشد تا بتواند یک دسته از نتایج را پردازش کند. برای تنظیم تعداد نتایجی که می خواهید برای هر دسته برگردانده شود، از پارامتر اندازه استفاده کنید.

• The search_after parameter

پارامتر `search_after` یک مکان‌نمای زنده ارائه می‌کند که از نتایج صفحه قبلی برای به دست آوردن نتایج صفحه بعدی استفاده می‌کند. این شبیه به عملیات اسکرول است که به معنای پیمایش بسیاری از پرس و جوها به صورت موازی است.

• Point in Time with search_after

نقطه در زمان (PIT) با `search_after` روش صفحه‌بندی ترجیحی در `OpenSearch` است، مخصوصاً برای صفحه‌بندی عمیق. محدودیت‌های همه روش‌های دیگر را دور می‌زند زیرا روی مجموعه داده‌ای کار می‌کند که در زمان ثابت است، به یک پرس و جو محدود نمی‌شود، و از صفحه‌بندی ثابت به جلو و عقب پشتیبانی می‌کند. برای کسب اطلاعات بیشتر، نقطه در زمان را ببینید.

2. Sort results

مرتب‌سازی به کاربران شما اجازه می‌دهد تا نتایج را به روشی که برای آنها معنادارتر است مرتب کنند. به طور پیش فرض، پرس و جوهایی متن کامل نتایج را بر اساس امتیاز مربوطه مرتب می‌کنند. شما می‌توانید با تنظیم پارامتر ترتیب به صعودی یا نزولی، نتایج را بر اساس هر مقدار فیلد به ترتیب صعودی یا نزولی مرتب کنید.

Search for a movie by title

Interstellar

Sort by

title

Sort order

desc

Search

Search results for: Interstellar (Sorted by title (desc))

Title: Moneyball and Interstellar

Director: Bennett Miller

Year: 2011

Title: Interstellar

Director: Christopher Nolan

Year: 2014

Delete Index

Autocomplete functionality .3

تکمیل خودکار پیشنهادهای را هنگام تایپ به کاربران نشان می دهد. برای مثال، اگر کاربری «pop» را تایپ کند، OpenSearch پیشنهادهای مانند «پاپ کورن» یا «popsicles» ارائه می کند. این آنها را سریع تر به یک عبارت جستجوی احتمالی هدایت می کند.

OpenSearch به شما امکان می دهد تکمیل خودکار طراحی کنید که با هر بار زدن کلید به روز می شود تکمیل خودکار را با استفاده از یکی از روش های زیرقابل پیاده سازی است:

- [Prefix matching](#)
- [Edge n-gram matching](#)
- [Search as you type](#)
- [Completion suggesters](#)
- **Prefix matching**

Movie Search App

Search for a movie by title

Star

Search

Search results for: Star

Title: Star Trek Beyond

Director: Justin Lin

Year: 2015

Enter a prefix for autocomplete suggestions

Int

Autocomplete suggestions for prefix 'Int':

```
▼ [
  0 : "Interstellar"
  1 : "Moneyball and Interstellar"
]
```

• Edge n-gram matching

در طول ایندکس سازی، n-gram یک کلمه را به دنباله ای از n کاراکتر تقسیم می کند تا از جستجوی سریع تر عبارات جستجوی جزئی پشتیبانی کند.

If you n-gram the word “quick,” the results depend on the value of n.

n	Type	n-gram
1	Unigram	[q, u, i, c, k]
2	Bigram	[qu, ui, ic, ck]
3	Trigram	[qui, uic, ick]
4	Four-gram	[quic, uick]
5	Five-gram	[quick]

Edge n-gramming the word “quick” results in the following:

- q
- qu
- qui
- quic
- quick

Movie Search App

Search for a movie

Search by

year

Search

Search results for: 20 by year

Title: Moneyball and Interstellar

Director: Bennett Miller

Year: 2011

Title: Interstellar

Director: Christopher Nolan

Year: 2014

Title: Star Trek Beyond

Director: Justin Lin

Year: 2015

Enter a prefix for autocomplete suggestions

Autocomplete by

year

Autocomplete suggestions for prefix '20' by year:

```
▼ [  
  0 : "2011"  
  1 : "2014"  
  2 : "2015"  
]
```

Completion suggerster •

پیشنهاد تکمیل: لیستی از پیشنهادات را می‌پذیرد و آنها را در یک مبدل حالت محدود (FST) می‌سازد، یک ساختار داده بهینه‌شده که اساساً یک نمودار است. این ساختار داده در حافظه زندگی می‌کند و برای جستجوی سریع پیشنوندها بهینه شده است. برای کسب اطلاعات بیشتر در مورد FST ها، به ویکی پدیا مراجعه کنید. همانطور که کاربر تایپ می‌کند، پیشنهاد دهنده تکمیل در نمودار FST هر بار یک کاراکتر در طول مسیر منطبق حرکت می‌کند. پس از تمام شدن ورودی کاربر، انتهای باقی مانده را بررسی می‌کند تا لیستی از پیشنهادات را تهیه کند.

Enter a prefix for autocomplete suggestions

Int

Autocomplete by

title

Autocomplete suggestions for prefix 'Int' by title:

```
▼ [
  0 : "Interstellar"
  1 : "Inception and Interstellar"
]
```

Search as you type •

Enter a prefix for autocomplete suggestions

20

Autocomplete by

year

Autocomplete suggestions for prefix '20' by year:

```
▼ [
  0 : "2008"
  1 : "2010"
  2 : "2014"
]
```

Did-you-mean .4

پیشنهاد دهنده آیا منظور شما اصلاحات پیشنهادی برای عبارت های جستجوی غلط املایی را نشان می دهد. به عنوان مثال، اگر کاربری "fliud" را تایپ کند، OpenSearch یک عبارت جستجوی اصلاح شده مانند "fluid" را پیشنهاد میکند. در بحث درسی ما Miss_Spelling است.

ما می توانیم پیشنهاد دهنده را با استفاده از یکی از روش های زیر پیاده سازی کنیم:

- برای پیشنهاد اصلاحات برای تک تک کلمات از پیشنهاد دهنده اصطلاح استفاده کنیم.
- از پیشنهاد دهنده عبارت برای پیشنهاد اصلاحات برای عبارات استفاده کنیم.

پیشنهاد اصلاحات برای تک تک کلمات:

Did-You-Mean Suggester

Enter search term:

Architecturee

Did you mean:

- architecture (score: 0.9166667)

پیشنهاد اصلاحات برای عبارات:

عبارت پیشنهاد دهنده شبیه اصطلاح پیشنهادگر است، با این تفاوت که از مدل های زبان n-gram برای پیشنهاد عبارات کامل به جای کلمات جداگانه استفاده می کند.

Did-You-Mean Suggester

Enter a search term:

design paterns

Did you mean:

design patterns (score: 0.31666178)

Did-You-Mean Suggester

Enter a search term:

Design Paterns

Did you mean:

- design **patterns** (score: 0.16) - Matches found

Candidate generators

Did-You-Mean Suggester

Enter a search term:

design paterns

Did you mean:

design patterns (score: 0.31666178)

Did-You-Mean Suggester

Enter a search term:

machine lerning

Top 1 suggestions for ' machine lerning ':

- machine learning (score: 0.04)

Keyword search

به طور پیش فرض، OpenSearch امتیازات اسناد را با استفاده از الگوریتم Okapi BM25 محاسبه می کند. BM25 یک الگوریتم مبتنی بر کلمه کلیدی است که جستجوی لغوی را برای کلماتی که در پرس و جو ظاهر می شوند انجام می دهد. هنگام تعیین ارتباط یک سند، BM25 فرکانس مدت/فرکانس سند معکوس (TF/IDF) را در نظر می گیرد:

فراوانی اصطلاح تصریح می کند که اسنادی که عبارت جستجو در آنها بیشتر دیده می شود، مرتبط ترند. بسامد معکوس سند به کلماتی که معمولاً در همه اسناد موجود در مجموعه ظاهر می شوند (به عنوان مثال، مقالاتی مانند "the") وزن کمتری می دهد.

Keyword Search in Shakespeare's Works

Enter search term:

long live king

Found 4 result(s):

- Play: Hamlet, Speaker: BERNARDO, Text: Long live the king! (Score: 0.61)
- Play: Richard III, Speaker: BUCKINGHAM, Text: Long live Richard, Englands royal king! (Score: 0.51)
- Play: Henry VI Part 2, Speaker: BOTH, Text: Long live our sovereign Richard, Englands king! (Score: 0.48)
- Play: Richard III, Speaker: GLOUCESTER, Text: Live long. (Score: 0.28)

Similarity algorithms

Algorithm	Description
BM25	The default OpenSearch Okapi BM25 similarity algorithm.
boolean	Assigns terms a score equal to their boost value. Use boolean similarity when you want the document scores to be based on the binary value of whether the terms match.

Specifying similarity

شما می توانید الگوریتم شباهت را در پارامتر شباهت هنگام پیکربندی نگاشتها مشخص کنید. به عنوان مثال، کوئری زیر شباهت بولی را برای `boolean_field` مشخص می کند. به `bm25_field` شباهت پیش فرض BM25 اختصاص داده شده است:

```
PUT /testindex
```

```
{
  "mappings": {
    "properties": {
      "bm25_field": {
        "type": "text"
      },
      "boolean_field": {
        "type": "text",
        "similarity": "boolean"
      }
    }
  }
}
```

Configuring BM25 similarity

```
PUT /testindex
```

```
{
  "settings": {
    "index": {
      "similarity": {
        "custom_similarity": {
          "type": "BM25",
          "k1": 1.2,
```

```

    "b": 0.75,
    "discount_overlaps": "true"
  }
}
}
}
}
}

```

BM25 similarity supports the following parameters.

Parameter	Data type	Description
k1	Float	Determines non-linear term frequency normalization (saturation) properties. The default value is 1.2.
b	Float	Determines the degree to which document length normalizes TF values. The default value is 0.75.
discount_overlaps	Boolean	Determines whether overlap tokens (tokens with zero position increment) are ignored when computing the norm. Default is true (overlap tokens do not count when computing the norm).

جستجوی مبتنی بر یادگیری ماشینی (ML)

• جستجوی برداری

▪ جستجوی k: k-NN-نزدیکترین همسایه یک عبارت جستجو را در میان شاخصی از بردارها جستجو می‌کند

k-NN search

افزونه k-NN که مخفف کلمه k-nearest همسایه است، کاربران را قادر می‌سازد تا k-نزدیک ترین همسایه ها را در یک نقطه پرس و جو در یک شاخص از بردارها جستجو کنند. برای تعیین همسایه‌ها، می‌توانید فاصله (تابع فاصله) را که می‌خواهید برای اندازه‌گیری فاصله بین نقاط استفاده کنید، مشخص کنید.

موارد استفاده شامل توصیه‌ها (به عنوان مثال، ویژگی «آهنگ‌های دیگری که ممکن است دوست داشته باشید» در یک برنامه موسیقی)، تشخیص تصویر، و تشخیص تقلب می‌شود. برای اطلاعات پس زمینه بیشتر در مورد جستجوی k-NN، به ویکی پدیا مراجعه کنید.

این افزونه از سه روش مختلف برای به دست آوردن k نزدیکترین همسایه ها از شاخص بردارها پشتیبانی می کند:

- **Approximate k-NN:**

روش اول از یک رویکرد تقریبی نزدیکترین همسایه استفاده می کند - از یکی از چندین الگوریتم برای برگرداندن تقریبی k-نزدیک ترین همسایه ها به بردار پرس و جو استفاده می کند. معمولاً این الگوریتم ها در ازای مزایای عملکردی مانند تأخیر کمتر، ردپای حافظه کوچک تر و جستجوی مقیاس پذیرتر، سرعت فهرست سازی و دقت جستجو را قربانی می کنند. برای کسب اطلاعات بیشتر در مورد الگوریتم ها، به مستندات nmslib و faiss مراجعه کنید. این الگوریتم بهترین انتخاب برای جستجوهای روی شاخص های بزرگ (یعنی صدها هزار بردار یا بیشتر) است که به تأخیر کم نیاز دارند. اگر می خواهید قبل از جستجوی k-NN، فیلتری را روی شاخص اعمال کنید، نباید از k-NN تقریبی استفاده کنید، که تعداد بردارهای مورد جستجو را به شدت کاهش می دهد. در این حالت باید از روش امتیازدهی اسکریپت یا پسوندهای Painless استفاده کنید. برای جزئیات بیشتر در مورد این روش، از جمله توصیه هایی برای استفاده از موتور، به جستجوی تقریبی k-NN مراجعه کنید.

- **Script Score k-NN:**

روش دوم عملکرد امتیازدهی اسکریپت OpenSearch را برای اجرای یک نیروی brute force، جستجوی دقیق k-NN در فیلدهای "knn_vector" یا فیلدی هابی که می توانند اشیاء باینری را نشان دهند، گسترش می دهد. با این رویکرد، می توانید جستجوی k-NN را بر روی زیرمجموعه ای از بردارها در فهرست خود اجرا کنید (گاهی اوقات به عنوان جستجوی پیش فیلتر نیز شناخته می شود). از این رویکرد برای جست و جو در بخش های کوچک تر اسناد یا زمانی که به یک فیلتر اولیه نیاز است، استفاده کنید. استفاده از این رویکرد در شاخص های بزرگ ممکن است منجر به تأخیر زیاد شود. برای جزئیات بیشتر در مورد این روش، دقیق k-NN با اسکریپت امتیازدهی را ببینید.

- **Painless extensions:**

روش سوم توابع فاصله را به عنوان اکستنشن های بدون درد اضافه می کند که می توانید در ترکیب های پیچیده تر استفاده کنید. مشابه امتیاز اسکریپت k-NN، می توانید از این روش برای انجام یک جستجوی brute force، k-NN دقیق در یک شاخص استفاده کنید، که از پیش فیلتر کردن نیز پشتیبانی می کند. این رویکرد در مقایسه با امتیاز اسکریپت k-NN عملکرد پرس و جو کمی کندتر دارد. اگر مورد استفاده شما

نیاز به سفارشی سازی بیشتری نسبت به امتیاز نهایی دارد، باید از این رویکرد در Script Score k-NN استفاده کنید. برای جزئیات بیشتر در مورد این روش، توابع اسکریپت نویسی بدون درد را ببینید.

به طور کلی، برای مجموعه داده‌های بزرگ‌تر، معمولاً باید روش تقریبی نزدیک‌ترین همسایه را انتخاب کنید زیرا مقیاس قابل توجهی بهتر است. برای مجموعه داده‌های کوچک‌تر، جایی که ممکن است بخواهید فیلتر اعمال کنید، باید رویکرد امتیازدهی سفارشی را انتخاب کنید. اگر مورد استفاده پیچیده‌تری دارید که در آن باید از تابع فاصله به عنوان بخشی از روش امتیازدهی آنها استفاده کنید، باید از رویکرد اسکریپت نویسی بدون درد استفاده کنید.

Exact k-NN with scoring script

PUT my-knn-index-1

```
{
  "mappings": {
    "properties": {
      "my_vector1": {
        "type": "knn_vector",
        "dimension": 2
      },
      "my_vector2": {
        "type": "knn_vector",
        "dimension": 4
      }
    }
  }
}
```

POST _bulk

```
{ "index": { "_index": "my-knn-index-1", "_id": "1" } }
{ "my_vector1": [1.5, 2.5], "price": 12.2 }
{ "index": { "_index": "my-knn-index-1", "_id": "2" } }
{ "my_vector1": [2.5, 3.5], "price": 7.1 }
```

```
{ "index": { "_index": "my-knn-index-1", "_id": "3" } }
{ "my_vector1": [3.5, 4.5], "price": 12.9 }
{ "index": { "_index": "my-knn-index-1", "_id": "4" } }
{ "my_vector1": [5.5, 6.5], "price": 1.2 }
{ "index": { "_index": "my-knn-index-1", "_id": "5" } }
{ "my_vector1": [4.5, 5.5], "price": 3.7 }
{ "index": { "_index": "my-knn-index-1", "_id": "6" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 10.3 }
{ "index": { "_index": "my-knn-index-1", "_id": "7" } }
{ "my_vector2": [2.5, 3.5, 5.6, 6.7], "price": 5.5 }
{ "index": { "_index": "my-knn-index-1", "_id": "8" } }
{ "my_vector2": [4.5, 5.5, 6.7, 3.7], "price": 4.4 }
{ "index": { "_index": "my-knn-index-1", "_id": "9" } }
{ "my_vector2": [1.5, 5.5, 4.5, 6.4], "price": 8.9 }
```

GET my-knn-index-1/_search

```
{
  "size": 4,
  "query": {
    "script_score": {
      "query": {
        "match_all": {}
      },
      "script": {
        "source": "knn_score",
        "lang": "knn",
        "params": {
```

```

    "field": "my_vector2",
    "query_value": [2.0, 3.0, 5.0, 6.0],
    "space_type": "cosinesimil"
  }
}
}
}}

```

خروجی کد:

Vector Search in OpenSearch

Enter vector search query for my_vector2 (comma-separated floats):

2.0, 3.0, 5.0, 6.0

Vector Search Results for Query Vector [2.0, 3.0, 5.0, 6.0]:

```

{
  "_index" : "my-knn-index-1"
  "_id" : "7"
  "_score" : 1.9995856
  "_source" : {
    "my_vector2" : [
      0 : 2.5
      1 : 3.5
      2 : 5.6
      3 : 6.7
    ]
    "price" : 5.5
  }
}

```

Vector Search in OpenSearch

Enter vector search query for my_vector2 (comma-separated floats):

2.0, 3.0, 5.0, 6.0

Vector Search Results for Query Vector [2.0, 3.0, 5.0, 6.0]:

```
{...}
{
  "_index" : "my-knn-index-1"
  "_id" : "6"
  "_score" : 1.9654887
  "_source" : {
    "my_vector2" : [
      0 : 1.5
      1 : 5.5
      2 : 4.5
      3 : 6.4
    ]
    "price" : 10.3
  }
}
```

2.0, 3.0, 5.0, 6.0

Vector Search Results for Query Vector [2.0, 3.0, 5.0, 6.0]:

```
{...}
{...}
{
  "_index" : "my-knn-index-1"
  "_id" : "9"
  "_score" : 1.9654887
  "_source" : {
    "my_vector2" : [
      0 : 1.5
      1 : 5.5
      2 : 4.5
      3 : 6.4
    ]
    "price" : 8.9
  }
}
```

```
2.0, 3.0, 5.0, 6.0
```

Vector Search Results for Query Vector [2.0, 3.0, 5.0, 6.0]:

```
{...}
{...}
{...}
{
  "_index" : "my-knn-index-1"
  "_id" : "8"
  "_score" : 1.9037901
  "_source" : {
    "my_vector2" : [
      0 : 4.5
      1 : 5.5
      2 : 6.7
      3 : 3.7
    ]
    "price" : 4.4
  }
}
```

All parameters are required.

- lang is the script type. This value is usually painless, but here you must specify knn.
- source is the name of the script, knn_score.

This script is part of the k-NN plugin and isn't available at the standard _scripts path. A GET request to _cluster/state/metadata doesn't return it, either.

- field is the field that contains your vector data.
- query_value is the point you want to find the nearest neighbors for. For the Euclidean and cosine similarity spaces, the value must be an array of floats that matches the dimension set in the field's mapping. For Hamming bit distance, this value can be either of type signed long or a base64-encoded string (for the long and binary field types, respectively).

- `space_type` corresponds to the distance function. See the [spaces section](#).

k-NN Search with Pre-filtering in OpenSearch

Enter vector search query (comma-separated floats):

9.9,9.9

Vector Search Results for Query Vector [9.9, 9.9]:

- Vector: [10, 10], Color: BLUE
- Vector: [20, 20], Color: BLUE

خروجی در open search چک شد.

```
200 - OK 30 ms
{
  "hits": {
    "total": {
      "value": 3,
      "relation": "eq"
    },
    "max_score": 0.98039204,
    "hits": [
      {
        "_index": "my-knn-index-2",
        "_id": "4",
        "_score": 0.98039204,
        "_source": {
          "my_vector": [
            10,
            10
          ],
          "color": "BLUE"
        }
      },
      {
        "_index": "my-knn-index-2",
        "_id": "5",
        "_score": 0.0048775724,
        "_source": {
          "my_vector": [
            20,
            20
          ],
          "color": "BLUE"
        }
      }
    ]
  }
}
```

k-NN Search with Binary Data and Hamming Distance in OpenSearch

Enter base64-encoded binary data:

Binary Search Results for Query Binary Data 23:

- Binary Data: TGFzdCBvbmUh, Color: BLUE
- Binary Data: SSBob3BIIHRoaXMgaXMgaGVscGZ1bA==, Color: BLUE

spaceType	Distance Function (d)	OpenSearch Score
l1	$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n x_i - y_i $	$score = \frac{1}{1 + d}$
l2	$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n (x_i - y_i)^2$	$score = \frac{1}{1 + d}$
linf	$d(\mathbf{x}, \mathbf{y}) = \max(x_i - y_i)$	$score = \frac{1}{1 + d}$
cosinesimil	$d(\mathbf{x}, \mathbf{y}) = 1 - \cos\theta = 1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\ \mathbf{x}\ \cdot \ \mathbf{y}\ }$ $= 1 - \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \cdot \sqrt{\sum_{i=1}^n y_i^2}}$ <p>where $\ \mathbf{x}\$ and $\ \mathbf{y}\$ represent the norms of vectors x and y respectively.</p>	$score = 2 - d$
innerproduct (supported for Lucene in OpenSearch version 2.13 and later)	$d(\mathbf{x}, \mathbf{y}) = -\mathbf{x} \cdot \mathbf{y} = -\sum_{i=1}^n x_i y_i$	<p>If $d \geq 0$,</p> $score = \frac{1}{1 + d}$
hammingbit	$d(\mathbf{x}, \mathbf{y}) = \text{countSetBits}(\mathbf{x} \oplus \mathbf{y})$	$score = \frac{1}{1 + d}$

Approximate k-NN search

جستجوی تقریبی k-نزدیکترین همسایگان (k-NN) تکنیکی است که در بازیابی اطلاعات و یادگیری ماشینی برای یافتن k نقاط در یک مجموعه داده که نزدیکترین نقطه به یک نقطه پرس و جو هستند، استفاده می‌شود. برخلاف k-NN دقیق، که فاصله‌ها یا شباهت‌ها را به طور کامل در تمام نقاط محاسبه می‌کند، k-NN تقریبی از تکنیک‌هایی استفاده می‌کند که دقت را فدای کارایی می‌کند. این مبادله به زمان‌های پرس و جو سریع‌تر اجازه می‌دهد، به ویژه در مجموعه داده‌های مقیاس بزرگ که جستجوی دقیق k-NN ممکن است از نظر محاسباتی گران شود.

Approximate k-NN Search in OpenSearch

Enter query vector (comma-separated floats):

2,3,5,6

Select knn_vector field:

my_vector2

Enter number of nearest neighbors to retrieve:

1

Approximate k-NN Search Results for Query Vector [2.0, 3.0, 5.0, 6.0] in Field 'my_vector2':

- Vector Field: [2.5, 3.5, 5.6, 6.7], Price: 5.5

The following table provides examples of the number of results returned by various engines in several scenarios. For these examples, assume that the number of documents contained in the segments and shards is sufficient to return the number of results specified in the table.

size	k	Number of primary shards	Number of segments per shard	Number of returned results, Faiss/NMSLIB	Number of returned results, Lucene
10	1	1	4	4	1
10	10	1	4	10	10
10	1	2	4	8	2

جستجوی عصبی:

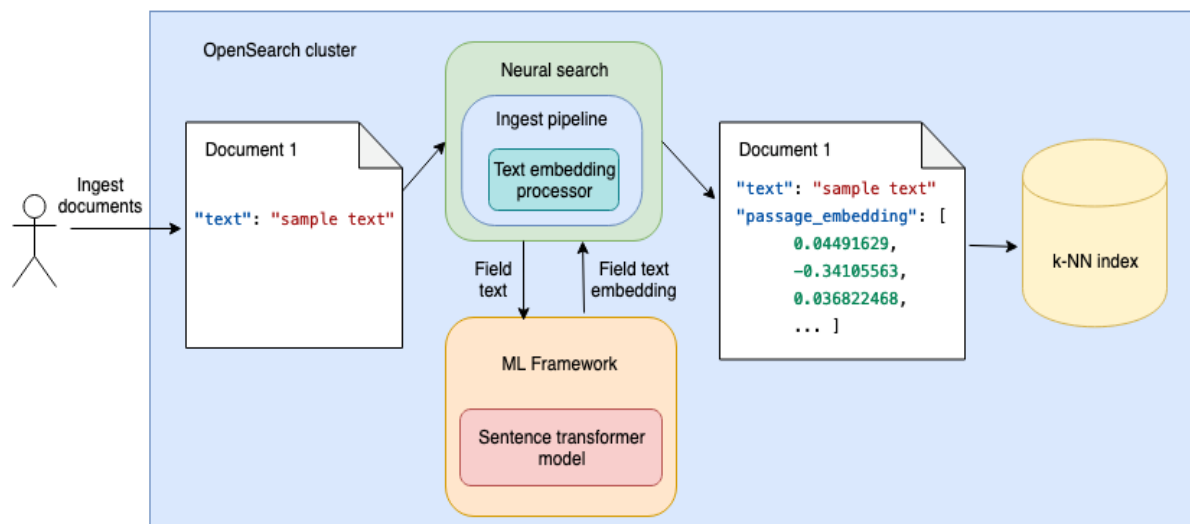
جستجوی عصبی تولید جاسازی‌های برداری را در زمان مصرف و جستجوی آنها در زمان جستجو را تسهیل می‌کند. جستجوی عصبی به ما امکان می‌دهد مدل‌های ML را در جستجوی خود ادغام کنیم و به عنوان چارچوبی برای اجرای سایر روش‌های جستجو عمل می‌کند. روش‌های جستجوی زیر بر روی جستجوی عصبی ساخته شده‌اند:

- **جستجوی معنایی:** معنای کلمات را در زمینه جستجو در نظر می گیرد. از بازیابی متراکم بر اساس مدل های جاسازی متن برای جستجوی داده های متنی استفاده می کند.
- **جستجوی چندوجهی:** از مدل های تعبیه چند وجهی برای جستجوی داده های متن و تصویر استفاده می کند.
- **جستجوی پراکنده عصبی:** از بازیابی پراکنده بر اساس مدل های جاسازی پراکنده برای جستجوی داده های متنی استفاده می کند.
- **جستجوی ترکیبی:** جستجوی سنتی و جستجوی برداری را برای بهبود ارتباط جستجو ترکیب می کند.
- **جستجوی مکالمه:** یک جستجوی مولد تقویت شده با بازیابی را پیاده سازی می کند.

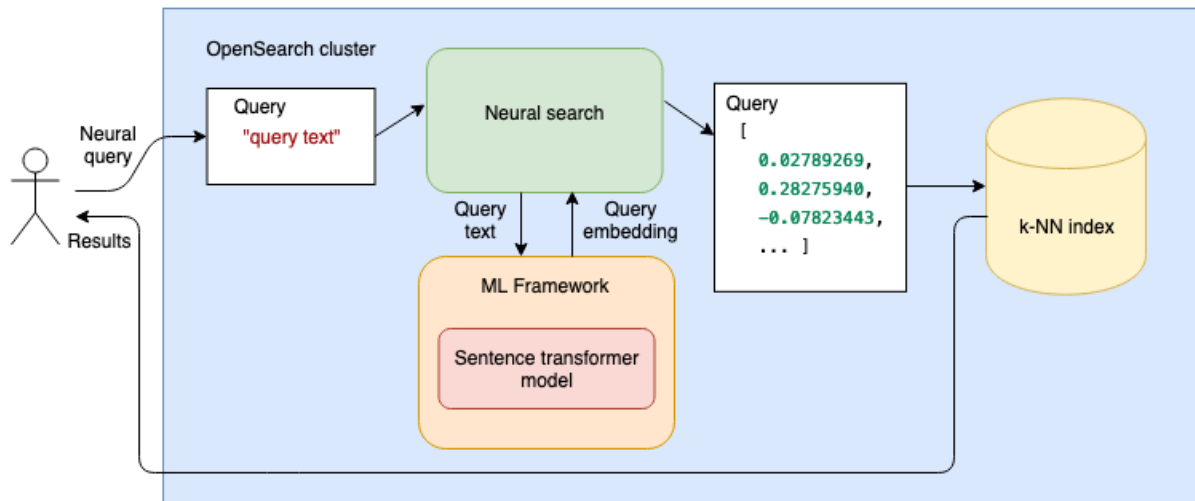
Terminology

Neural search: Facilitates vector search at ingestion time and at search time:

- At ingestion time, neural search uses language models to generate vector embeddings from the text fields in the document. The documents containing both the original text field and the vector embedding of the field are then indexed in a k-NN index, as shown in the following diagram.



- At search time, when you then use a *neural query*, the query text is passed through a language model, and the resulting vector embeddings are compared with the document text vector embeddings to find the most relevant results, as shown in the following diagram.



- Semantic search*: Employs neural search in order to determine the intention of the user's query in the search context, thereby improving search relevance.
- Hybrid search*: Combines semantic and keyword search to improve search relevance.

سرچ عصبی به صورت عملی:

در ادامه با اجرای مسیر زیر سرچ عصبی انجام خواهیم داد:

Set up an ML language model

```
import pandas as pd
from opensearchpy import OpenSearch
import streamlit as st
```

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('intfloat/multilingual-e5-large')
```

Adding Documents from Dataset

```
df = pd.read_csv('imdb_top_1000.csv')
```

Creating vectors

```
df["Vector"] = df["Series_Title"].apply(lambda x:
model.encode(x))
```

Creating indexing

```
client = OpenSearch(
    hosts=[{'host': 'localhost', 'port': 9200}],
    http_auth=('admin', 'admin')
)

index_name = 'imdb_movie'

def create_index_with_knn_vector(client, index_name):
    if client.indices.exists(index=index_name):
        client.indices.delete(index=index_name)

    index_body = {
```

```
"settings": {
  "index.knn": True
},
"mappings": {
  "properties": {
    "vector": {
      "type": "knn_vector",
      "dimension": 1024
    },
    "Series_Title": {
      "type": "text"
    },
    "Released_Year": {
      "type": "text"
    },
    "Runtime": {
      "type": "keyword"
    },
    "Genre": {
      "type": "text"
    },
    "IMDB_Rating": {
      "type": "float"
    },
    "Overview": {
      "type": "text"
    },
    "Director": {
      "type": "text"
    },
    "Star1": {
      "type": "text"
    },
    "Star2": {
      "type": "text"
    },
    "Star3": {
```



```

        "type": "text"
    },
    "Star4": {
        "type": "text"
    },
    "No_of_Votes": {
        "type": "integer"
    }
}
}

response = client.indices.create(index=index_name,
body=index_body)
create_index_with_knn_vector(client, index_name)

```

indexing Documents

```

def index_documents():
    for idx, row in df.iterrows():
        document = {
            'Series_Title': row['Series_Title'],
            'vector': row['Vector'],
            'Released_Year': row['Released_Year'],
            'Runtime': row['Runtime'],
            'Genre': row['Genre'],
            'IMDB_Rating': row['IMDB_Rating'],
            'Overview': row['Overview'],
            'Director': row['Director'],
            'Star1': row['Star1'],
            'Star2': row['Star2'],
            'Star3': row['Star3'],
            'Star4': row['Star4'],
            'No_of_Votes': row['No_of_Votes']
        }
        response = client.index(index=index_name,
body=document)
index_documents()

```

NLP For Query

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('intfloat/multilingual-e5-large')

index_name = 'imdb_movie'
client = OpenSearch(
    hosts=[{'host': 'localhost', 'port': 9200}],
    http_auth=('admin', 'admin')
)

def search_movies(query):
    query_vector = list(model.encode(query))
    query = {
        "size": 5,
        "query": {
            "script_score": {
                "query": {"match_all": {}},
                "script": {
                    "source": "knn_score",
                    "lang": "knn",
                    "params": {
                        "field": "vector",
                        "query_value": query_vector,
                        "space_type": "cosinesimil"
                    }
                }
            }
        }
    }

    response = client.search(index=index_name, body=query)
    return response

def main():
    st.title('IMDb Movie Search')
    query = st.text_input('Enter your search query:')
```

```

if st.button('Search'):
    results = search_movies(query)
    st.subheader('Search Results:')

    st.markdown("""
<style>
.card {
    border: 1px solid #ddd;
    border-radius: 10px;
    padding: 10px;
    margin: 10px 0;
}
</style>
""", unsafe_allow_html=True)

    for hit in results['hits']['hits']:
        st.markdown(f"""
<div class="card">
    <p><b>ID:</b> {hit['_id']}</p>
    <p><b>Title:</b>
{hit['_source']['Series_Title']}</p>
    <p><b>Overview:</b>
{hit['_source']['Overview']}</p>
    <p><b>IMDB Rating:</b>
{hit['_source']['IMDB_Rating']}</p>
    <p><b>Director:</b>
{hit['_source']['Director']}</p>
</div>
""", unsafe_allow_html=True)

if __name__ == '__main__':
    main()

```

با استفاده از این دو خط کوعری را به بردار تبدیل می‌کنیم:

```
From sentence_transformers import SentenceTransformer
model = SentenceTransformer('intfloat/multilingual-e5-large')
```

با استفاده از `cosinesimilarity` درصد شباهت بردار کوعری با داکيومنت را در پلتفرم مورد نظر بدست آورده و مرتبط ترین سرچ برگردانده می‌شود.

خروجی:

IMDb Movie Search

Enter your search query:

car

Search

Search Results:

ID: 7OixU5ABPUpYNMxXkgwJ

Title: Drive

Overview: A mysterious Hollywood stuntman and mechanic moonlights as a getaway driver and finds himself in trouble when he helps out his neighbor.

IMDB Rating: 7.8

Director: Nicolas Winding Refn

ID: JOixU5ABPUpYNMxXFQug

Title: Casino

Overview: A tale of greed, deception, money, power, and murder occur between two best friends: a mafia enforcer and a casino executive compete against each other over a gambling empire, and over a fast-living and fast-loving socialite.

IMDB Rating: 8.2

Director: Martin Scorsese

ID: xeixU5ABPUpYNMxXPgvB

Title: Baby

Overview: An elite counter-intelligence unit learns of a plot, masterminded by a maniacal madman. With the clock ticking, it's up to them to track the terrorists' international tentacles and prevent them from striking at the heart of India.

IMDB Rating: 8.0

Director: Neeraj Pandey

ID: UeixU5ABPUpYNMxX-w4K

Title: Planes, Trains & Automobiles

Overview: A man must struggle to travel home for Thanksgiving with a lovable oaf of a shower curtain ring salesman as his only companion.

IMDB Rating: 7.6

Director: John Hughes

ID: huixU5ABPUpYNMxXwA0h

Title: Crash

Overview: Los Angeles citizens with vastly separate lives collide in interweaving stories of race, loss and redemption.

IMDB Rating: 7.7

Director: Paul Haggis