elastic 中国开发者大会 2023

Elastic Stack 新功能在各种业务场景下的应用

朱杰 资深解决方案架构师
Elastic 2023/4

1 基础架构

2 搜索场景

3 可观测场景

4 安全场景

5 未来展望

6 解决方案

1. 基础架构

基础架构方面改进

7.0	查询时智能选择副本 跨集群查询来回通讯最小化 没有查询的分片跳过写入刷新
7.3	仅参与投票的主节点
7.4	通过API Key来进行认证 自动取消进行中的查询 更小的Translog
7.6	跨集群查询、跨集群复制的代理模式
7.10	简化了节点角色的定义方式
7.12	ES Kibana 支持ARM
7.15	节点间通讯流量可以选择开启压缩
7.16	增强集群承载能力，处理更多分片

8.0	简化了全栈的安全配置
8.3	降低Master 和 Data 节点堆内存消耗 查询旧版本ES创建的索引
8.5	改进GET by _id 性能 bloom filter改进
8.6	新的集群平衡策略

查询旧版本ES创建的索引

8.3

使用可搜索快照的功能

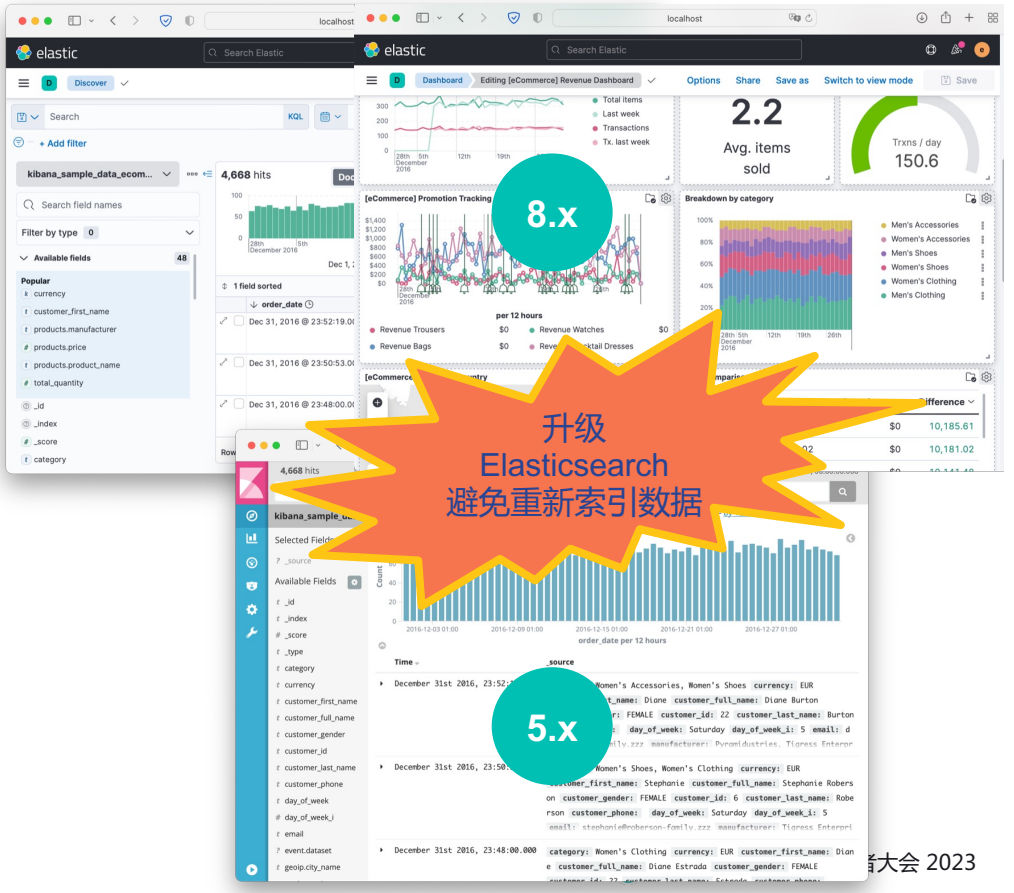
对于全观测和安全场景比较有用

支持的数据类型: numeric, keyword, text,

boolean, date, object, ip, etc.

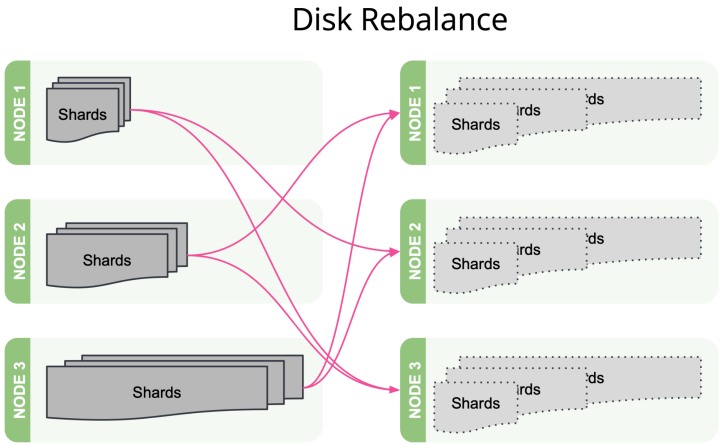
在集群中就像一个常规的索引

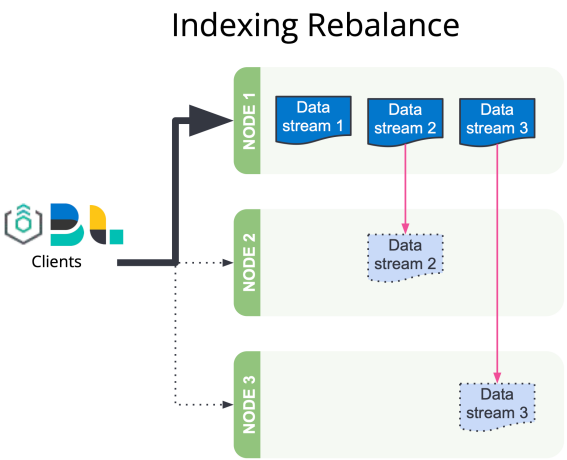
减少升级到新版本的障碍



新的集群平衡策略 8.6

- **目的**
  - 对于大的集群，分片不均匀是长期以来的痛点
- **功能**
  - 避免某些节点磁盘热点触发水位线
  - 避免某些索引热点节点
  - 改善集群资源利用和稳定性
  - 减轻集群运维人员负担
- **两种策略**:
  - 根据分片大小对磁盘进行再平衡
  - 根据写入数据流负载对索引进行再平衡





增强集群扩展性系统优化项目 7.16

经过大量测试和分析，发掘各个系统影响扩展性的问题

Fix Large Shard Count Scalability Issues #77466

Open · 69 of 88 tasks · original-brownbear opened this issue on Sep 9, 2021 · 3 comments

original-brownb... commented on Sep 9, 2021 • edited by DaveCTurner — Member

This meta issue tracks known issues with scaling clusters to large numbers of shards.

- Security
  - Improve Authorization performance in clusters with a large number of indices #67987
    - Authorization for Internal Requests like Stats or Rollover is Slow in Large Clusters #79632
- General
  - Push back on excessive requests for stats #51992
  - Add pagination to diagnostic APIs #87555
  - Make Cache More Memory Efficient #77546
  - Cluster Stats API Slows down Considerably for Larger Clusters #79563
  - Reduce merging in PersistedClusterStateService #79793
  - Make org.elasticsearch.action.admin.cluster.state.ClusterStateResponse Compress the Cluster State #79906
  - optimize getIndices in IndicesSegmentResponse #80064
  - Massive async shard fetch requests consume lots of heap memories on master node. #80694
  - Pending task batching can be a bottleneck #81626
  - Batch add-block-index-to-close, add-index-block and finalize-index-block tasks #81627
  - Stop unnecessary retries of shard-started tasks #81628
  - Batch up master tasks to create, mount, and delete snapshots #81846
  - Batch up failure-related ILM master tasks #81880
  - Stats actions should discard intermediate state on cancellation #82337
  - `RestClusterGetSettingsAction` Requests the Full Metadata from Master #82342
  - More Compact Serialization of Metadata #82608
  - Make TaskBatcher Less Lock-Heavy #82227
  - Speed up MappingStats Computation on Coordinating Node #82830
  - Add level=datastreams to Indices Stats #83049
  - (Re)Starting a Data Node Holding a Large Number of Indices can Take Minutes #83203
  - A Node Joining a Cluster with a Large State Receives the Full Uncompressed State in a ValidateJoinRequest #83204 -> Reduce resource needs of join validation #85380
  - RecoverySourceHandler#runWithGenericThreadPool caused deadlock #85839
  - Report stats related to new sizing guidance #86639
  - Make GetIndexAction cancellable #87681
  - Drop `ClusterStateHealth#indices` when unnecessary #90631
  - Improve scalability of BroadcastReplicationActions #92902
  - Store Disk Threshold Ignore Setting in IndexMetadata #78672
  - Speed up Routing Nodes Priority Comparator #78609
  - Allow indices lookup to be built lazily #78745
  - Optimize XContent Object Parsers #78813
  - Find a way to Deduplicate Index Settings #78892
    - Implement setting deduplication via String interning #80493
  - MasterService#patchVersions is rather inefficient #77888
  - Replace RoutingTable#shardsWithState(...) with RoutingNodes#unassigned(...) #78931
  - Speedup computing cluster health #78969
  - IndexMetadataUpdater#applyChanges is rather inefficient #78980
  - Batch Cluster State Updates in Datastream Rollover #79782
  - Batch Index Settings Update Requests #79866
  - Make MasterService.patchVersions not Rebuild the Full CS #79860
  - Save some RoutingNodes Instantiations #79941
  - Cache `DiscoveryNode#trimTier` Result #80179
  - Rework ILM to not Require Inspecting all Indices on every Cluster State Update #80407
  - Batch up failure-related ILM master tasks #81880
  - Faster ShardsLimitAllocationDecider #82251
  - Large ILM Task Batches are Executed too Slowly #82708
  - Make AllocationService#adaptAutoExpandReplicas Faster #83092
  - Speed up Building Indices Lookup in Metadata #83241
  - Speed up Name Collision Check in Metadata.Builder #83340
  - Make LIFECYCLE_NAME_SETTING a Field in IndexMetadata #83582
  - Use static empty store files metadata #84034
  - `MetadataIndexAliasesService` submits unbatched tasks at URGENT priority #89924
- Search
  - Group shard request per node in the field capabilities API #74648
  - Group shard request per node in the can match phase #78164
  - Intern IndexFieldCapabilities Type String on Read #76405
  - Fix NumberFieldMapper Referencing its Own Builder #77131
  - Fix MatchOnlyTextFieldMapper Retaining a Reference to its Builder #77201
  - Fix TextFieldMapper Retaining a Reference to its Builder #77251
  - Search Responses to Many Shards use Excessive Amounts of Memory for OriginalIndices instances #78314
  - Filter original indices in shard level request #78508
  - Merge field caps responses on each node? #82879
  - Updating index metadata does an expensive validation of the mapping even if unchanged #89309
  - Searches against a large number of unavailable shards result in very large responses #90622
- Network
  - Sending Large Transport Messages Should be Optimized #82245
  - APIs like /_cluster/state Break for Large Clusters due to Response Size Limitations #79560
  - Render Mappings more Compact in GET /_cluster/state #83846
  - Add the Ability to Disable certain REST APIs via a Cluster Setting #84876
  - Track distribution of REST response sizes #84887
  - Make use of chunked REST response infrastructure in more APIs #89838

支持ARM体系架构

7.12 Elasticsearch Kibana 正式支持ARM

20%
性能提升

10%
成本下降

AWS Graviton2 和 X86实例比较

开发都喜欢……Apple M1 M2

2. 搜索场景

搜索场景的改进

7.0	Block Max Wand算法改进根据score的排序
7.2	根据距离的权重和script score 边输入边搜索，搜索建议
7.3	Interval query改进 wildcard prefix规则 扁平字段类型
7.4	Result Pin 设置搜索结果保持置顶 Script score的向量距离函数
7.6	Block Max Wand算法改进根据时间和数值排序
7.7	时间排序的查询性能进一步优化
7.9	Wildcard字段类型
7.10	基于时间基线的查询 Wildcard字段不分大小写查询
7.13	Combined字段类型
7.14	Term枚举API
7.16	按照时间排序的search_after性能增强

8.0	Lucene9 向量搜索
8.2	Range query性能增强 Lookup来加入额外字段，有限的join支持 随机采样聚合运算
8.3	Geo grid查询
8.4	搜索API支持传统query和knn一起查询和
8.5	向量搜索GA

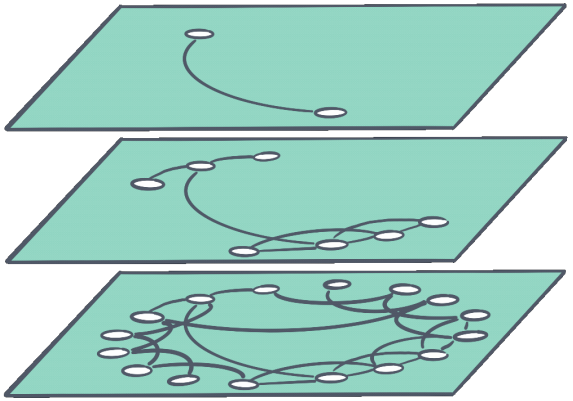
全新的向量搜索 8.5

7.3版本使用暴力搜索的方式

- 在有很多过滤条件和Top的小数据量下可以工作
- 在大的索引上查询延迟很大

8.0使用HNSW & KNN

- 降低一点精确性获得大规模扩展能力
- 在大规模的索引上也有好的性能



```
POST products/_search
{
  "query": {
    "multi_match": {
      "query": "black flower dress",
      "fields": ["title", "description"],
      "boost": 0.9
    }
  },
  "knn": {
    "field": "title_vector",
    "query_vector": [0.3, 0.1, ...],
    "k": 5,
    "num_candidates": 50,
    "boost": 0.1
  },
  "size": 20
}
```

Lookup 运行时字段 8.2

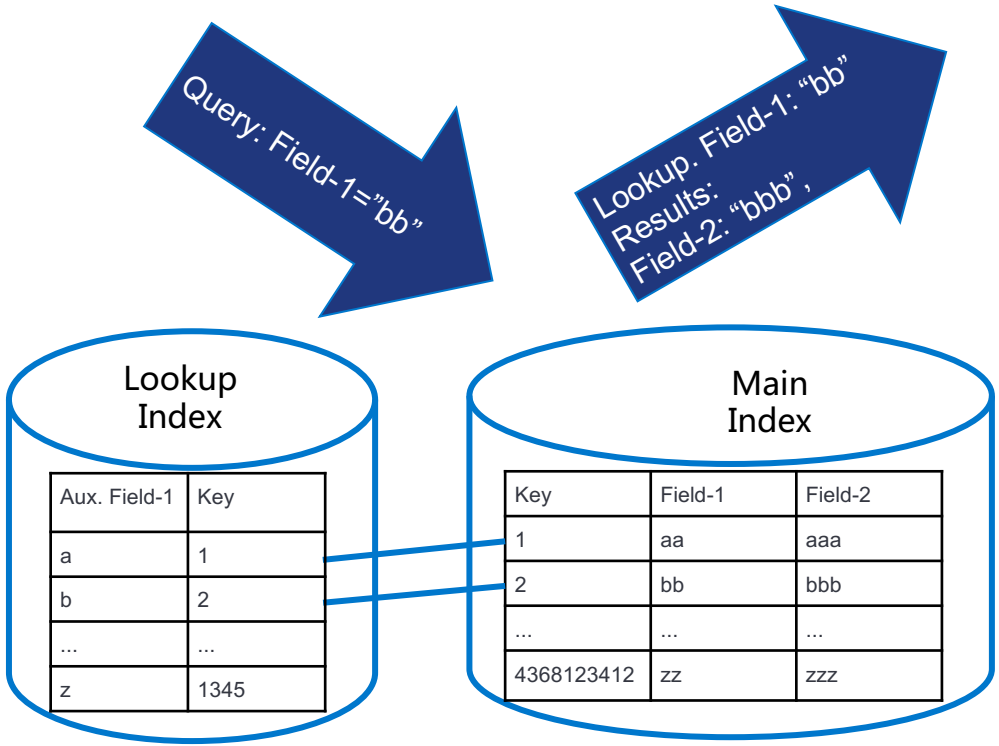
并不是 Join!

lookup的字段不能进行过滤和聚合

和写入时的lookup类似, 但是是用在查询结果处理

应用场景:

- 有些被join的字段更新比较频繁
- 在返回结果前补全某些字段



基于时间基线的搜索 7.10

- 数据在不停写入，refresh在自动执行，后台merge也在自动进行，Segment会不停变化
- 但是业务需求希望前后查询能保持稳定
- Search_after 翻页的时候希望保持前后一致
- 需要让ES保持住某个时间点Segment的状态，不能被merge掉
- 要注意资源消耗

PIT请求　保持时间

```
POST /my-index-000001/_pit?keep_alive=1m
```

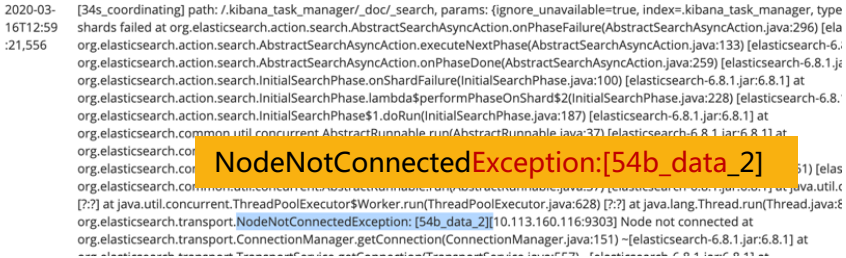
```
POST /_search 1
{
    "size": 100,
    "query": {
        "match" : {
            "title" : "elasticsearch"
        }
    },
    "pit": {
            "id":  "46ToAwMDaWR4BXV1aWQxAgZub2RlXzEAAAAAAAAAAAEBYQNpZHkFdX
            "keep_alive": "1m"  3
    }
}
```

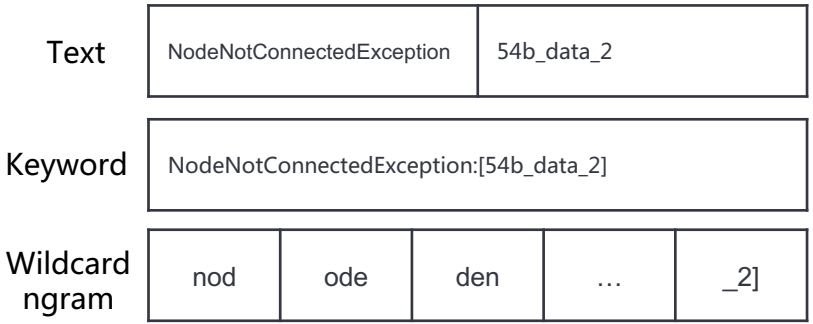
查询需要携带前面的ID

```
DELETE /_pit
{
    "id" : "46ToAwMDaWR4BXV1aWQxAgZub2RlXzEAAAAAAAAAAAEBYQNpZHkFdXVpZDIrBm
}
```

删除PIT

Wildcard 字段 7.9





- 在日志和安全场景，很多时候需要**任意字符串的查询**，分词会比较困难
- Keyword有长度限制
- 通配符开头的查询速度很慢

字符串的n-gram	进行粗糙的第一遍搜索
原始字符串的doc value	从第一遍结果中精确匹配

- 简化了构造搜索表达式
- 简化了配置分词器
- 比使用通配符查询更快的速度

优化搜索排序

机器生成的数据往往需要根据时间排序

机器生成的数据往往体量非常大

通过block max wand算法跳过尽量跳过没有竞争力的文档

需要total hits和聚合计算场景无效

大幅优化
_score 排序

大幅优化
Date Number
排序

进一步优化
Date Number
排序，跳过不
包含符合条件
的整个分片

大幅优化
Date Number
排序后的
search_after

地理位置搜索聚合

- 7.4 Shape字段类型
- 7.5 Geotile composite 聚合
- 7.6 地理位置搜索性能优化
- 7.8 更好的地理位置聚合
- 7.11 使用Geo_line聚合来寻找路线
- 7.12 对geo_point进行geo_shape 查询
- 7.14 Runtime field通过长宽查询geo shape
  对geo_shape进行geotile grid 聚合性能提升15%
- 8.3 Geo grid query

3. 可观测场景

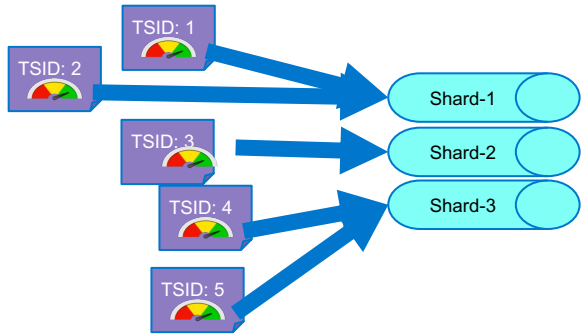
可观测场景

7.6	Block Max Wand算法改进根据时间和数值排序
7.7	异步搜索 Lucene数据结构放到堆外 时间排序的查询性能优化
7.9	Wildcard字段类型
7.10	Stored Field压缩算法改进
7.12	Runtime Field, Schema on read
7.13	可搜索快照GA
7.14	Match Only Text字段类型
7.16	增强集群扩展性，能放更多分片 时序化索引的Segment sorter

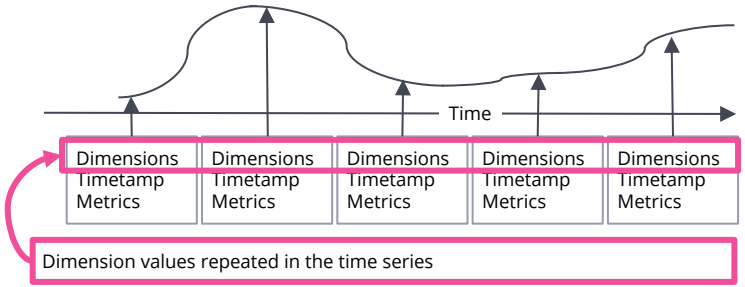
8.1	字段支持仅保存Doc Value
8.4	从Doc Value合成_source
8.6	TSDB 时序数据优化

TSDB 时序数据优化 8.6

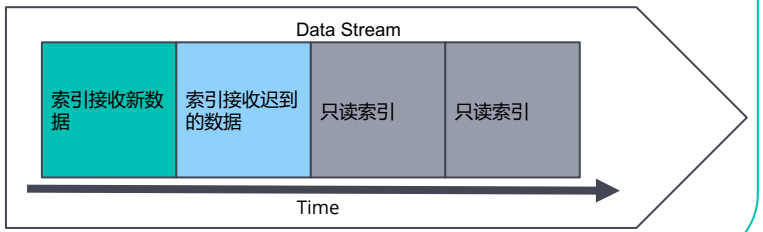
根据TSID进行路由



根据 TSID & timestamp排序

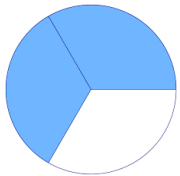


不同的索引时间戳不会交错



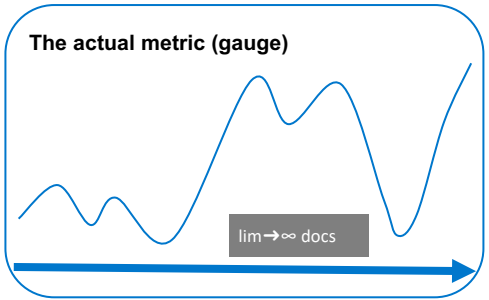
降低约~44%索引大小

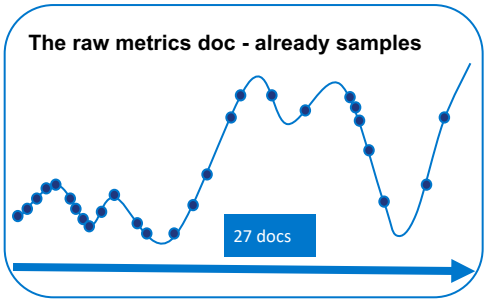
- 降低的程度不同的索引会不一样

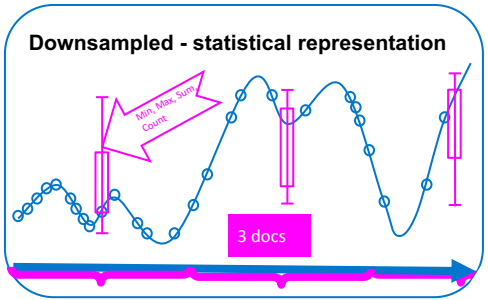


TSDB降采样 8.6

- 降采样是生命周期管理的任务
- 降采样到一个新的索引
- 容易配置和管理
- 全面支持Kibana分析组件Lens, TSVB, Agg-based vis, Timelion
- 对data strea查询的时候
  - 简单对data strea进行查询
  - 会自动包含原始索引和多个降采样索引一起查询







仅保存Doc Value的字段 8.1

减少>20% 存储空间
提升>20% 构建索引吞吐

没有倒排索引和BKD索引(index: false)

不影响聚合计算性能 (doc_values: true)

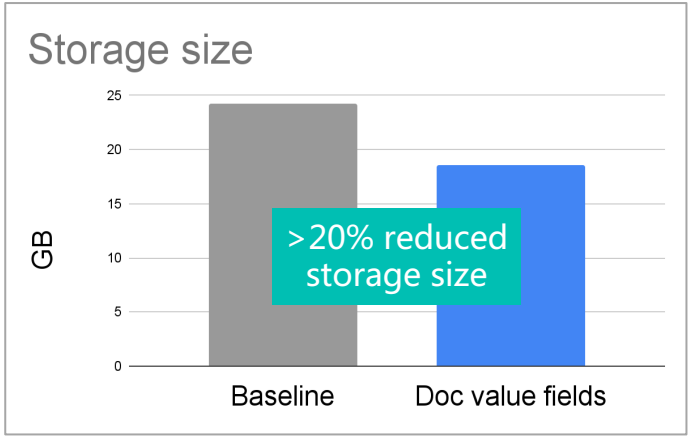
查询性能会慢 ~5-10 倍

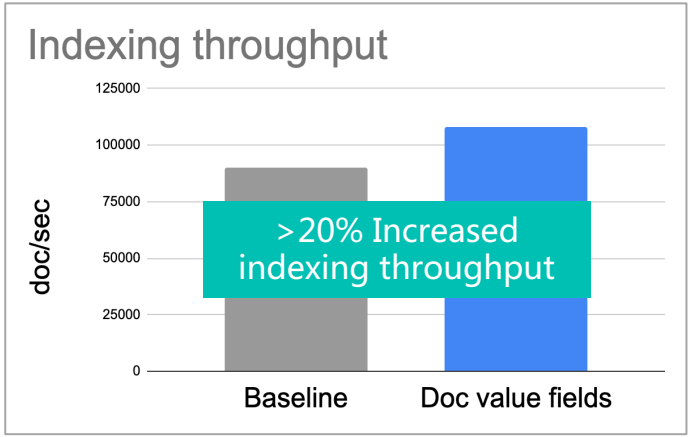
理想应用场景:

- 某些字段只是用来聚合计算
- 其它字段用来查询和过滤
- 指标数据

涉及的数据类型:

- Numeric
- Date
- Keyword
- IP
- Geo_point
- Boolean





从Doc Values合成_source

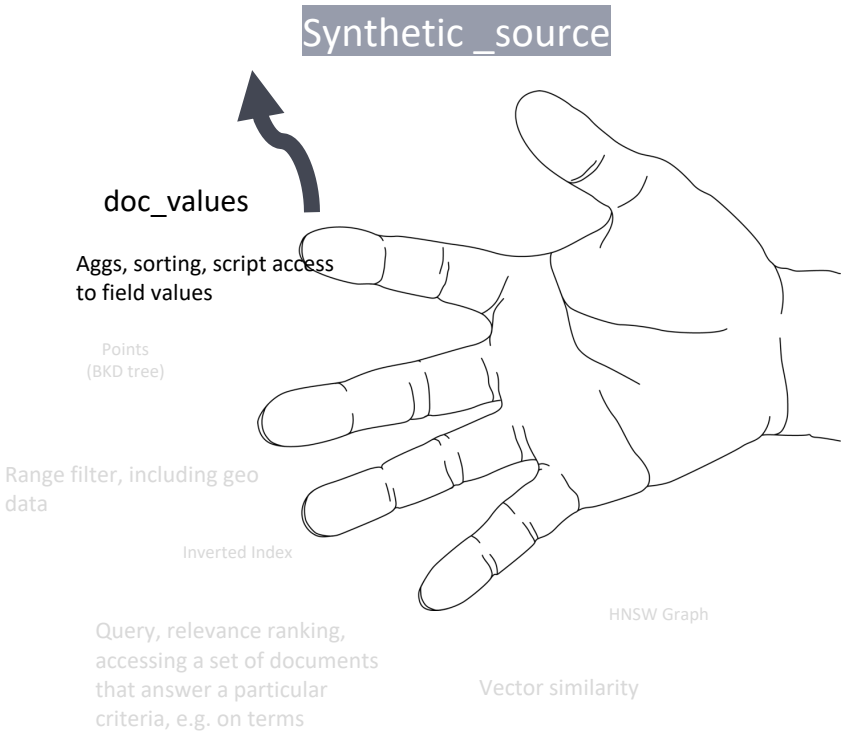
8.4

优势:

- 减少索引大小
- 提高写入性能

限制:

- 有时候轻微降低 get _source 性能
- 支持的数据类型
  boolean, byte, double, float, geo_point, half_float, integer, ip, keyword, long, scaled_float, short
- Runtime fields不能使用合成_source
- _source 的字段顺序可能改变



Runtime Field 7.12

在mapping中定义

```
PUT /test {
 "mappings": {
   "properties": {
     "@timestamp": {
       "type": "date",
       "format": "strict_date_optional_time||epoch_second"
     },
     "message": {
       "type": "wildcard"
     },
     "status": {
       "type": "runtime",
       "runtime_type": "long",
       "script": "String m = doc[\"message\"].value; int end =
m.lastIndexOf(\" \"); int start = m.lastIndexOf(\" \", end -
1) + 1; emit(Long.parseLong(m.substring(start, end)));"
     }
   }
}
```

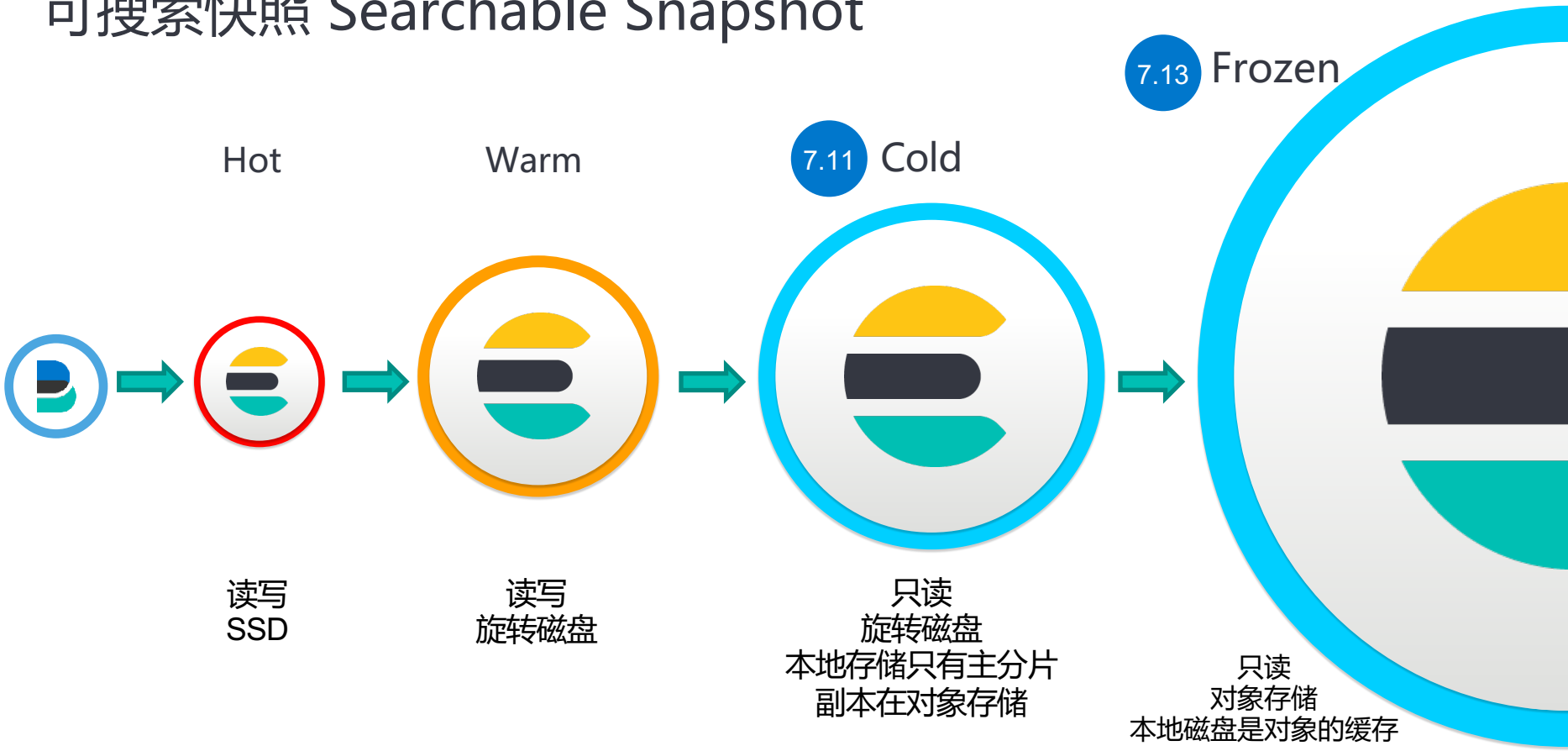
在单个查询中定义

像其它字段一样进行查询&聚合

```
POST /_async_search
{
  "runtime_mappings": {
     "ip": {
       "type": "runtime",
       "runtime_type": "ip",
       "script": "String m = doc[\"message\"].value;
emit(m.substring(0, m.indexOf(" ")));"
     }
  },
  "query": {
    "bool": {
      "must": [
        { "range": { "ip": { "gte": "40.135.0.0", "lt":
"40.135.255.255" } } },
        { "match": { "status": "200" } },
        { "range": { "@timestamp": { "gte": "1998-05-01T00:00:00Z",
"lt": "1998-05-02T00:00:00Z" } } }
      ]
    }
```

```
{
  "timestamp": "1998-04-30T14:30:17-05:00",
  "message" : "40.135.0.0 - - [1998-04-30T14:30:17-05:00] \"GET /images/hm_bg.jpg HTTP/1.0\" 200 24736"
}
```

可搜索快照 Searchable Snapshot



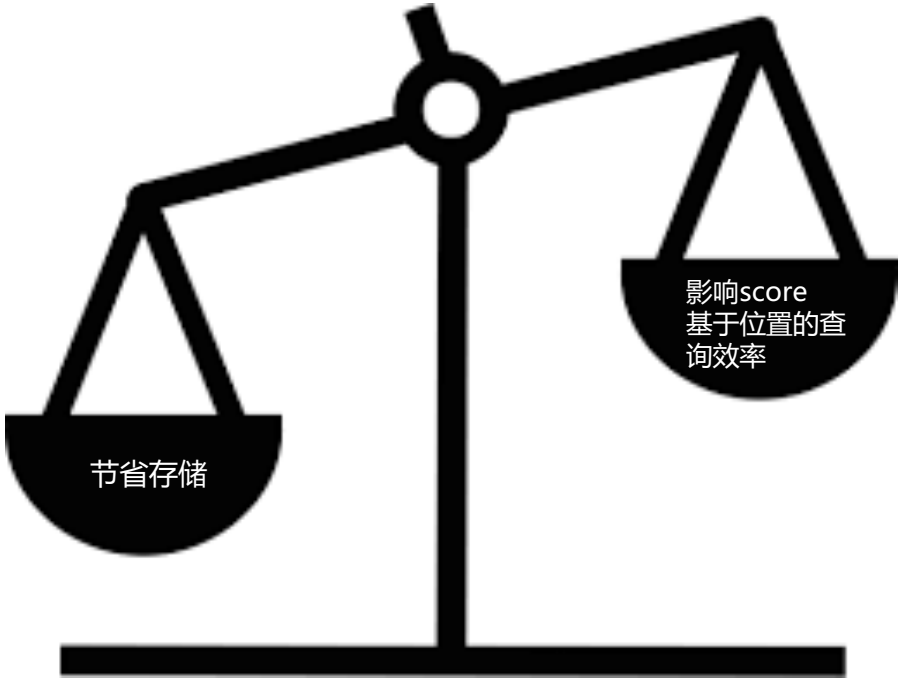
Match Only Text 7.14

专门为日志准备的字段类型，代替Text类型

日志搜索排序一般基于时间戳排序，而不是基于相关性

可以不存储跟相关性排序的数据
- Length normalization factors
- Term frequencies
- Positions

不支持Span查询，支持其它text字段支持的查询
Phrase 和 Intervals查询比text字段慢



全新的集中化数据采集平台

Elastic Agent

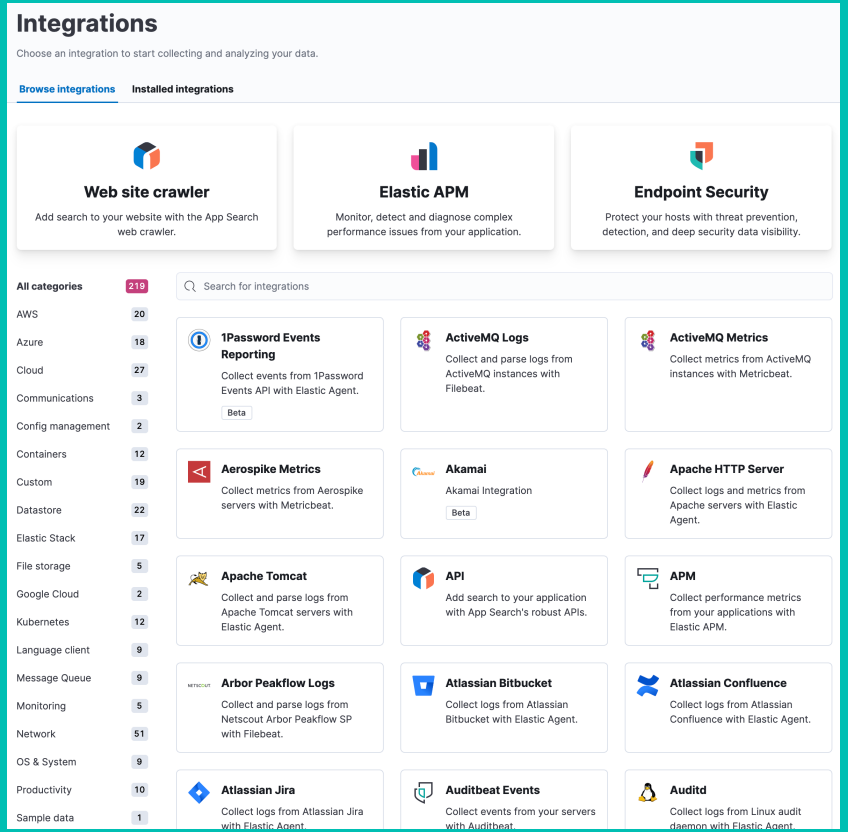
单一的Agent，统一的方式来采集日志、指标
还有其它类型的数据。
统一的配置方式，方便集成各种数据源

Fleet

Agent集中化管理平台
方便大规模管理Agent
可以集中下发、更新配置，不再需要依赖第三方工具

Integration

在Kibana中提供UI来集成各种服务
陆续会支持100+ Beats模块
不仅接入不同数据源，还提供开箱即用的Dashboard、可视化图表、数据处理Pipeline



4. 安全场景

EQL 事件查询语言

EQL用来从一个时间序列中找到符合条件的事件

- 多个事件有先后发生顺序的约束
- 有事件之间时间间隔的约束
- 不同数据源需要根据某个键值join
- 需要符合过滤条件

- 安全检索场景有大量复杂事件查询的需求，而Elasticsearch标准查询无法完成这样的搜索
- 源自Endgame的Python实现，用Java重写并集成进ES
- 7.9版本发布时性能测试 基于MITRE 2019 dataset 400万事件 3.9GB json文件 查询平均20秒以下，比python版本快20倍
- 7.16版本得益于Segment Sort和 search_after性能提升，有进4倍性能提升
- 7.16优化join key的null处理，一些场景有最大800倍提升

8:00:00 process日志记录中一个名叫regsvr32.exe的进程运行，process.id=900

8:05:00 文件操作行为日志 file中 process.id=900的进程加载了一个scrobj.dll的文件

两个事件必须发生在1小时以内

```
sequence with maxspan=1h
  [ process where process.name == "regsvr32.exe" ] by process.id
  [ file where file.name == "scrobj.dll" ] by process.id
 until [ process where event.type == "termination" ]
```

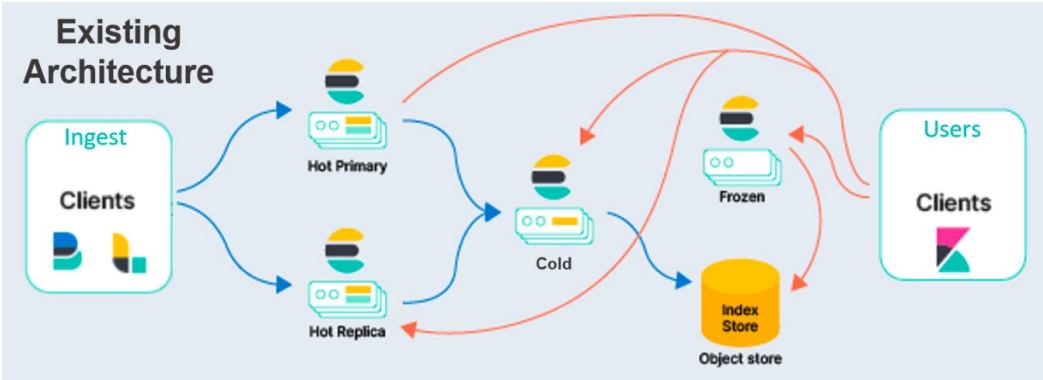
遇到条件提前结束

关联键

5. 未来展望

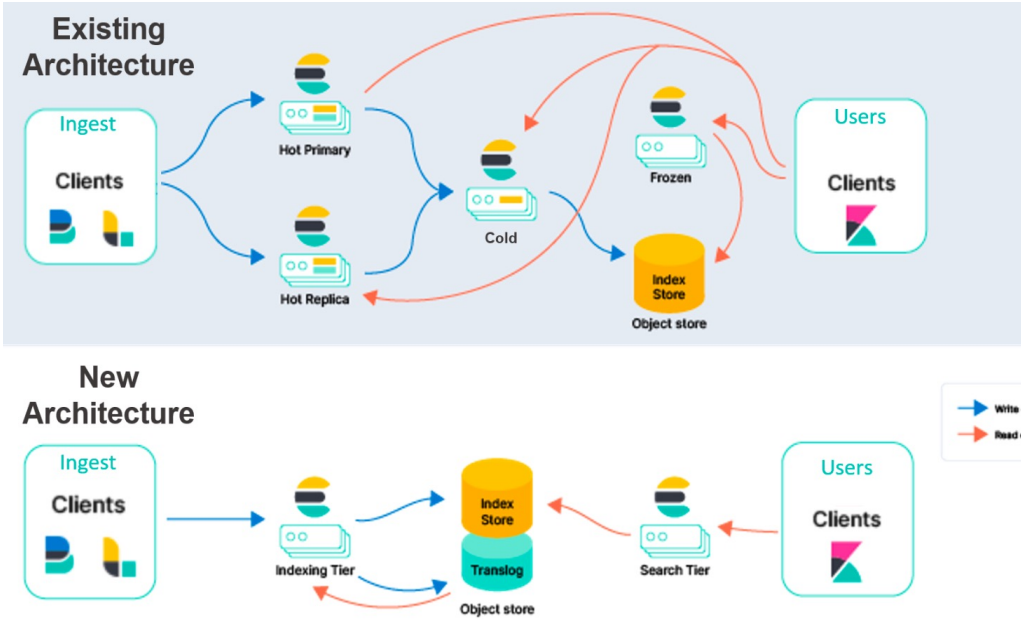
云原生Serverless架构要解决的问题

- 写入损耗，副本也需要消耗cpu资源构建索引
- 多份数据拷贝
- 写入和搜索共享CPU
- 冷热分层，加大规划难度
- 大量数据移动



云原生Serverless架构

- 存算分离
- 写入和搜索分离，只需要创建一次索引
- 可以独立扩展每一层
- 使用对象存储获得高可用和海量存储
- 消除数据复制
- 使用本地缓存来提高性能
- 自动伸缩，甚至计算资源到0



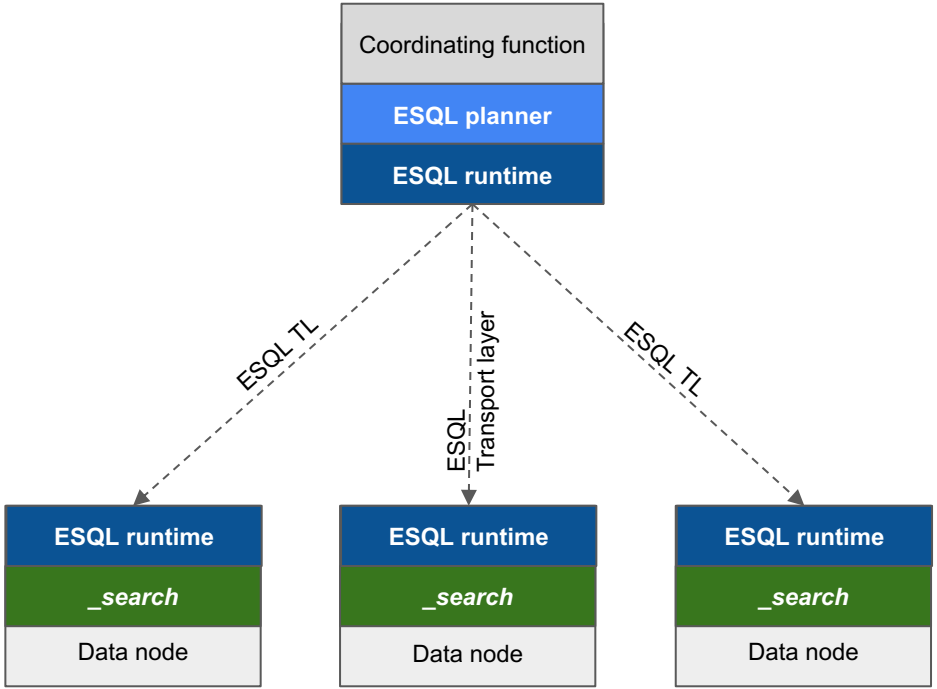
ESQL 新的查询语言

- 搜索 →
  聚合/转换 →
  输出
- 管道
  - Lookups
  - Joins
  - Subqueries

```
from logs-endpoint
| where event.category == "file"
| stats filecount = count(file.name) by process.name,host.name
| eval split = process.name | dissect split "%{process}.%{extension}"
| eval fullproc = concat(process, ".", extension)
| eval proclength = length(process.name)
| sort filecount,proclength desc
| limit 10
| project host.name,process.name,filecount, process, extension, fullproc
```

ESQL 计算引擎

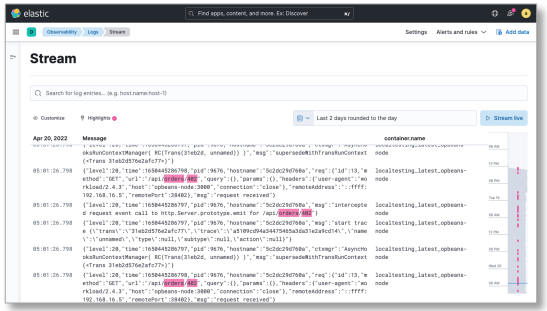
- Parse -> Analyze -> Optimize -> Execute
- 支持大数据量
- 未来扩展
  - 外部数据源
  - 其它语言的客户端



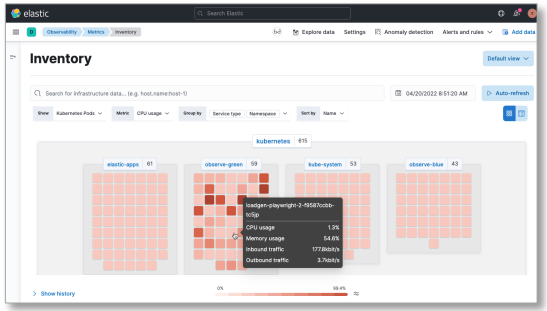
6. Elastic解决方案更新

全观测解决方案演进

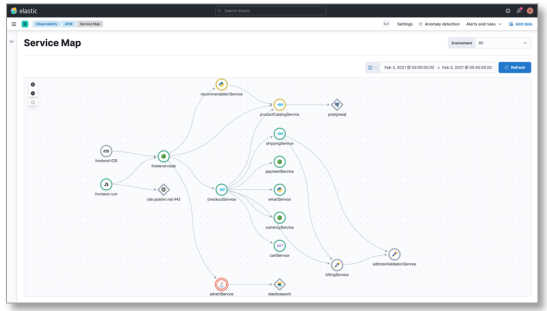
日志



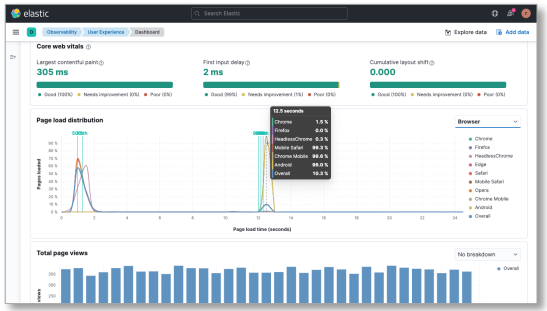
指标



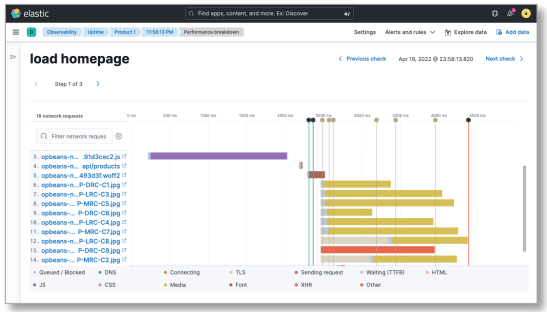
分布式链路分析



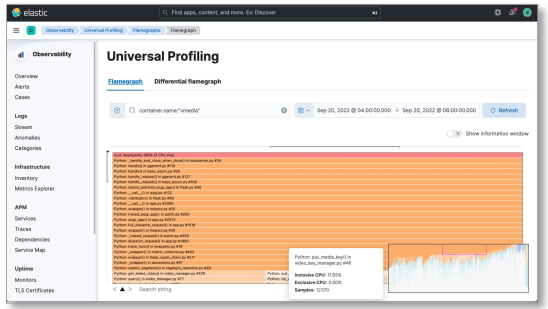
真实用户监测



模拟监测



性能剖析



安全解决方案

主动防御

执行前防御

- ❑ 恶意软件防御
- ❑ 勒索软件防御
- ❑ MBR勒索软件防御
- ❑ 内存威胁防御

执行后防御

- ❑ 勒索行为防御
- ❑ 恶意行为防御

信息采集

持续检测

- ❑ Kernel级别数据采集
- ❑ 主机数据采集
- ❑ 基于osquery的数据采集

持续检测

- ❑ 告警处置和威胁捕获工作流
- ❑ 洞察，上下文，推荐
- ❑ 集成威胁情报
- ❑ 预制检测规则: 案例, 规则, 机器学习模型
- ❑ 高级分析, 交互式可视化和根因分析
- ❑ 快速和可扩展的搜索平台，开放的数据格式定义
- ❑ 支持自部署和多云环境

安全响应

- ❑ 调查和响应工作流
- ❑ 和外部告警系统集成: email, Slack, SOAR & ITSM 平台
- ❑ 外部 case 连接器: IBM, JIRA, ServiceNow, Swimlane
- ❑ 用户自定义连接器

- ❑ 执行osquery检查
- ❑ 远程操作主机隔离

elastic 中国开发者大会 2023

感谢观看

专业、垂直、纯粹的 Elastic 开源技术交流社区

https://elasticsearch.cn/