

# 开源与云 Elasticsearch应用剖析

• 集结10位资深大咖 • 讲透4大技术场景 • 分享3大云服务应用场景



阿里巴巴集团副总裁贾扬清与Elastic 创始人&CEO  
共话“开源共生,云聚未来”,与社区大咖分享最前沿  
的技术应用与案例。





钉钉扫码加入  
“Elasticsearch 技术交流群”



扫码了解  
“更多阿里云 Elasticsearch”



扫码订阅  
“阿里云 Elasticsearch 技术博客”



阿里云开发者“藏经阁”  
海量免费电子书下载

# | 目录

云生态下的开源共生之路	4
云上 Elasticsearch 驱动新未来	8
Elastic Stack 解决方案与行业应用	15
阿里云 Elasticsearch 云原生内核	27
如何规划和执行威胁狩猎	39
基于流式计算平台搭建实时分享	50
Elasticsearch 基于 Pipeline 窗口函数	63
Elasticsearch 在清博大数据的应用与实践	73
Elasticsearch 在企查查的应用实践	83
Elasticsearch 在乐言的多场景应用实践	88

# 云生态下的开源共生之路

摘要：众所周知，阿里云与 Elastic 已成功合作三年，在这三年的合作过程中，双方在开源体系、云原生和数字化转型等领域中，都有各自不同的理解和洞察。本篇内容将向大家展开阿里巴巴集团副总裁贾扬清与 Elastic 创始人兼 CEO Shay Bannon，对当下热门技术领域和云生态下开源共生之路的探讨。

分享人：阿里巴巴集团副总裁贾扬清，Elastic 创始人兼 CEO Shay Bannon

视频地址：<https://developer.aliyun.com/live/246146>

2017 年，阿里云携手 Elastic 合作推出了阿里云 Elasticsearch 服务，并一举获得成功，成为影响全球的服务。据统计，目前数千个阿里云用户都在使用这项服务，包括很多传统型用户，支持数据已经超过 10PB。相信在三年的成功合作后，阿里云与 Elastic 的合作仍会继续。

在这样的成功合作背景下，阿里巴巴集团副总裁贾扬清和 Elastic 创始人兼 CEO Shay Bannon 视频连线，共同探讨了对当下热门技术领域和云生态下开源共生之路。

## 一、云原生的技术开放性

Elastic 可以帮助用户获取数据。如果用户需要在网站、应用、移动设备或基础设备商增加搜索功能，以便进行运营监控；或需要添加 Elastic Stack 以防御外部的各种威胁，Elastic 就是专注于解决这些问题的好选择。

随着云原生的功能逐步落地，搜索已成为核心技术以及根基所在。因此，开发的产品必须要简便易用，易于管理，秉承开发者第一、API 第一的原则，用户借助 API 及优秀的 UI 体验，可以轻松地开展业务，并插入到业务流程工具和系统中。所以 Elastic 所提供的服务可以赋能云原生产品，用户可以使用 Elastic 和阿里云合作推出的产品服务来赋能自身的业务服务，实现一键式的部署和扩容。

Elastic 与阿里云团队的合作，将 Elastic 的能力、构架和经验落地，带给用户良好的

云原生体验。

#### 具体案例：

有很多用户都在使用 Elasticsearch，用户们都能够在使用后快速提升自身的业务发展，比如老师和学生的连接更高效，教学质量提升。他们不需要经常地扩大 IT 团队或基础设施，云原生的赋能作用就能够快速将大量设备和服务进行整合，使客户能把重心放在自身业务上。

## 二、云原生对开源的影响

首先是开源的适用性，开源往往与开发者的本处构想是一致的。云原生的很多产品对基础架构来说起到引领和基础的作用，比如 Elasticsearch 的 Kibana 和 Kubernetes，都是开源的架构。基于这些开源架构有助于建设更好的基础建设，又能进一步赋能云原生的各项功能。

## 三、开源的视角

开源对云来说影响也是很大的。过去有很多的界面他们的 API 是封闭的，并且互不协作，所以如果想要建立端到端的系统比如搜索，数据库等功能，就需要大量基础性工作。开源体系让不同的成分可以轻松地互相连接；而且开源永远以开发者为中心，业务的搭建通过技术可以变成一个简便易用的架构体系，使得业务高效运转。

从云提供商的角度和开源的角度来看待事物，可以帮助我们从基础层面打破之前的技术障碍，可以拥有更开放的标准和公共技术平台，这样就能够推动此类技术在过去 20 年中人们所建设的基础设施和架构层面进行部署。

阿里巴巴有专门的开源技术委员会，并希望进一步地为开源社区作出贡献。阿里有大量的开发者，在进行超过 1000 个已经对外发布的开源项目；而且实际上，在 GitHub 上大概有约 70 万开发者在时刻关注和跟进。所以，阿里巴巴正在积极地回馈开源社区。

另外，阿里巴巴也在研究和解决大型系统项目方案拥有丰富的经验，以此推动开源社区发展。开源社区非常强调以用户为中心，以及以用户体验为中心。

## 四、企业数字化转型

**Shay Bannon 先生：**

Elastic 开发了一些产品，赋能企业的 CIO 和领导以最快的速度进行业务部署。数字化转型的特点包括：第一，越来越多的业务在数字化，比如现在远程工作的时候聊天应用或共享文档取代了电话联系。而 Shay Bannon 先生相信任何的数字化都应该是可搜索的，每个人在公司都能触达这样的功能，以便做出更好的商业决策。所以搜索应该是发现技术流动的基础所在，而且可以确信的是，搜索本身也是在数字化转型中所孕育的。这也适用于客户的产品，比如将某些业务转入到网上，变成手机上的应用，那么就可能需要一个搜索框帮助进行信息的搜索，这是链接整个系统最有效的方式。

很明显，推动数字化转型需要越来越多基础设施上的职称，更关键的是，数字化转型的目的是业务能否真正地为客户提供服务，所以产生了 Observability 这个产品组合。

最后就是安全。随着越来越多的数字足迹产生，可能会面临更多的攻击，必须要能发现并排除他们，这时通过数据搜索发现完全问题就尤为重要。如果需要确保得到有效的保护，就需要 Elastic 安全产品的帮助，这也是目前世界范围内大量组织的选择，特别是产品内有一个专门的安全设置叫做 SIEM。

Shay Bannon 先生确信世界范围内，搜索已经是无处不在，应用广泛，过去几年搜索的确成为了通用的用户体验。这也意味着未来能打造更好的搜索产品，因为通过大量的应用能够建立反馈机制，及时了解用户新的需求，并针对需求采取行动。相关性也是非常重要的，所以 Elastic 建立的大量相关模型，而这些模型都是建立在开源体系之上。Elastic 与开源社区的合作以及在全球建立的社群使 Elastic 始终保持在领先地位，这也是来自于成千上万的开发者帮助产品日日更新，实时进步。Elastic 把自己定义为一家搜索业务公司，因此必须确保搜索功能真正发挥其核心作用。

**贾扬清先生：**

数字化转型需要大量高性能机器以确保整体的稳定性，而且未来几年大量用户也会经历类似的变化。最大的变化就是出现人工智能和大数据的融合，这也和所预测的一样。



## 五、阿里云与 Elastic 的三年合作和未来规划

对 Shay Bannon 先生来说，事情的出发点一定是人。过去几年阿里巴巴的合作非常高效，共同组织活动，开发更好的产品，确保提供更大的价值给到阿里云的客户和 Elastic 的客户。相信双方可以在整体架构上搭建任何东西，合作非常愉快。

不止于此，Shay Bannon 先生表示，能把 Elastic 的产品和力量带到中国这个充满市场并将其实实在在地部署到整个阿里云。双方诸多方面的合作帮助他们的产品跨过界限，让 Elastic 认识到与阿里云的合作对业务发展起到至关重要的作用。

双方的合作已经步入第四年，贾扬清先生在此分享了两件事。

第一，基于合作，双方已经打造了多条非常行之有效、可复制的路径促进开源和云原生方面的合作。

另外一点是前沿技术，在这方面是有非常优秀的人才来负责的。其他方面在业务端进行深度合作，用技术服务客户，阿里一直在思考开源经济，Stack Overflow 创建者曾经发表过一份关于如何实现开源，以及如何通过技术和业务来推动开源的发展的公开信。

贾扬清先生表示，双方在重建传统的生态系统和尝试相互进行技术和业务的补足上是一个很成功的例子。双方也有关于测试路径上的争论，目的是如何共同面向市场，不仅在中国，未来在全球都能合作共赢。这些过往的合作经验指引着双方走向更美好，更有成就的未来。

Shay Bannon 先生表示，阿里云和 Elastic 已经建立了坚实的基础，开发了很多新的产品和服务，帮助用户能够快速使用。这样的合作非常棒，如果能够创建一个商业搜索产品，或者让 Elastic 的 Observability 产品更高效，这样用户就能对于在云基础设施进行监控，并进一步提升安全能力。所有这些努力会让搜索更快、更相关，并且融合了人工智能，所有的运行情况都能实时在阿里云的服务端得到反馈。

基于阿里云与 Elastic 三年的合作成果，相信双方在未来以“开源共生、云聚未来”的核心理念不变的情况下，将为开发者带来更多有意义、有价值的产品能力与体验。

# 云上 Elasticsearch 驱动新未来

摘要：开源最大的特征就是开放性，云生态则让开源技术更具开放性与创造性，Elastic 与阿里云的合作正是开源与云生态共生共荣的典范。值此合作三周年之际，我们邀请业界资深人士相聚云端，共话云上 Elasticsearch 生态与技术的未来。

分享人：阿里巴巴集团资深技术专家邓万禧

视频地址：<https://developer.aliyun.com/live/246147>

本篇主要通过 2 个部分介绍云上 Elasticsearch 如何驱动新未来：

- 阿里云 Elasticsearch 三周年回顾
- 新产品发布

## 一、阿里云 Elasticsearch 三周年回顾

阿里云 Elasticsearch 在 2017 年云栖大会由阿里云和 Elastic 公司共同发布。在产品设计之初，阿里云作为产品开发团队就坚持源于开源而不止于开源的理念，希望能够给客户 提供百分百兼容开源生态和软硬件一体化设计的云原生产品，降低企业上云的成本，提升效率。

随着云原生的发展和云服务的发展，我们认为应该从更广义的角度去理解云原生的概念。因云而产生的软件、硬件、架构，就是真正的云原生。从企业客户的角度，云原生的服务需要能够给客户带来极致的弹性和完善的运维能力，通过上云能够给企业带来降本增效的收益，而阿里云 ES 就是通过长期大规模的云上实践而发展起来的云原生产品。

我们主要围绕以下三个方向进行了探索，主要包括：

- Elastic 开源技术站和阿里云基础设施的高效集成；
- 软硬一体化的架构演进；
- 场景化内核的增强。



以下是产品发展过程中的几个关键性指标和 milestone。



2017 年，阿里云 ES 发布商业化的 Elastic Stack5.5 版本，全部架构都基于云原生的基础设施，包括 ECS、VPC、SRB 和云盘等等，完全兼容 ES 的 API。此产品主要从云管控的角度解决企业客户，多集群的运维效率低的痛点问题，比如一键拉起、平滑升级等等。希望在云上的客户可以更高效的方式使用和运维自己的 Elasticsearch 服务。

2018 年，主要工作的重点是发挥阿里云基础设施上的优势，覆盖更多的区域和更多的产品形态进行输出。通过部署 Elasticsearch 的集群提升了 ES 服务的稳定性，也希望借助于异构的存储架构，降低用户在实际场景的存储成本。同时阿里云发布了智能运维产品 EYou1.0，进一步提升了多集群的运维效率。

2019 年，我们主要实现了 ES 全栈的组件托管，具备全栈的云原生能力，并且演进为计算存储分离的架构，为进一步提升弹性扩缩能力奠定基础。另外，也降低了热节点存储的成本，提升了一些性能，尤其是在时序和日志场景。

2020 年，我们对云原生服务能力进行了升级，发布了智能弹性伸缩和 DevOps 相关的能力，另外也包括自研的报警监控服务等。

通过以上几个服务，阿里云团队进一步打磨了产品，并且也继续深入到业务场景中，持续优化业务的架构，为云上客户带来更好的产品体验。

经过三年多的发展，技术架构经过多轮的迭代和演进，阿里云 ES 已经发展成为覆盖开源 ES 全栈组件的云原生产品。

下图展现了阿里云 ES 的整体产品架构，从上到下包括场景化的能力，内核，和平台架构三部分。



## 1. 场景化能力，包括多模态的搜索，可观测性，和安全分析。

### 1) 多模态搜索

阿里云 ES 集成了达摩院的向量检索、内核和阿里 NLP 分词，增强了对文本和图片的理解能力，并且针对用户 DB 加速场景的需求，我们也针对性地做了多项改进，大大提升了在订单，交易等场景的稳定性。

### 2) 可观察性

针对海量日志存储和写入成本问题，发布了阿里云日志增强版本，不仅时序场景的性能达到 150%，并且降低了 40% 的成本。从数据前链路上，云原生的 beats 数据采集，从产品层面和阿里云 ACK 无缝打通，可以方便地采集 coordinates 日志做安全审计。另外，高级监控服务基于 Elasticsearch 作为存储实现了独立的一个产品，通过更细粒度和更高精度的 matrix，极大地提升了 ES 集群的监控能力，提升了运维效率。

### 3) 安全分析

除了针对 Ad-hoc 查询做了定制优化之外，我们也和阿里安全团队合作，对安全场景的可视化能力做了增强，方便用户使用。

## 2. 内核相关的增强

我们的策略是尽可能地跟进社区的版本，为云上的客户带来最新版本的能力。另外，也结合阿里云的基础设施对内核做了增强，我们也希望能够把通用的部分贡献给社区，为开源社区发展贡献我们的一份力量。

下面列举一些比较重要的改进给大家分享。

### 1) 通用物理复制

主要效果为写入性能提升了 70%，它的原理主要是在 Elasticsearch 的主节点做索引构建，而副本只读，主节点构建完成之后利用高效的网络分发 section 到副本，节省了索引构建的计算消耗。

### 2) 阿里云 QoS 插件

由于查询侧针对 DB 加速的场景对稳定性的高要求，我们发布了阿里云 QoS 插件，基于阿里云智能引擎的长期实践，我们结合自研的网络管理组件 Dig，实现了对查询和节点级别更精准的流控和降级能力。它可以在网络基础设施抖动的情况下，对查询的影响降到最低。

### 3) Fast Bulk

另外一个显著上的优化称之为 Fast bulk，可提升写入性能接近 30%。

以上就是部分优化的一些例子，也希望有更对同学对内核感兴趣的话与我们交流。

## 3. 云管控平台

在架构层面，我们基于阿里云存储和计算的基础设施，完成了多次的架构迭代。首先在高可用上，我们完成了一键的多可用区部署，通过多可用区的高可用，更好地实现了高可用的能力。我们主要解决的问题是一键部署和秒级切流，让用户可以在出现故障的情况下快速地完成业务切流，把影响降到最低。第二，基于 hot-warm 架构和计算存储分离架构，我

们大大地降低了存储成本，让用户在同等花销下存储更多的业务数据。除此之外，基于共享存储，在 20 年上半年我们也发布了基于 ES 的智能伸缩特性，主要适用场景是业务高峰和低谷期差异特别明显的场景，比如直播和交通出行，解决的问题是业务低峰期资源浪费的痛点问题。

当然，为了更好地提升集群的运维效率，我们也一直在迭代智能运维系统 EYou，通过 EYou 的持续迭代提升用户多集群的管理和运维效率。

总结一下，从整体上看，我们后续会持续从场景化，内核和架构的三个方向去持续迭代产品。具体的技术方向包括等级点的成本优化，自研引擎内核的集成和持续场景的写入吞吐的优化，后面在产品发布上也会和大家分享这方面工作的最新进展。

从业务规模上看，阿里云 ES 已经覆盖国内和国外 17 个区域，数据量达到了数十 PB，集群数也达到了数万以上的规模。从场景上看，我们服务了包括在线教育，新零售，游戏等多个行业的互联网和传统企业，也希望在今后的时间能够继续服务好云上的客户，给整个行业的客户赋能。

## 二、新产品发布

### 1. 产品简介

这个新产品的功能称之为 Indexing Service，它将会在今年 2 月份全新上线，主要适用于时序场景，给云上客户降低机器成本，提升运维效率。从客户角度上看，Indexing Service 的特性包括以下三个部分：



- 解决时序数据的高并发写入瓶颈，性能提升 150%。
- 优化集群写入的计算成本，热节点的成本降低 70%。
- 降低运维复杂度，它是一个云原生的 Serverless 产品形态，可以实现秒级的弹性扩缩，让用户更少地关注集群的运维

## 2. 技术架构

下面介绍的是 Indexing Service 的技术架构。



首先从适用场景上来看，它适用于日志，监控，和 APM 等时序场景。它解决的痛点问题是指集群写多读少的场景时，机器成本和运维成本高的问题。它的使用方式也非常简单，首先按照吞吐和 qps 实际需求开通按量付费，在产品控制台上可以一键开启，无需做业务上的改进，完全兼容原生的 API。而随着业务流量的增加，Indexing Service 可以做到秒级的快速弹性伸缩。最后，从内部实现上看，Indexing service 是读写分离的架构，对写的性能做了针对性的优化。

## 3. 内部流程

下面从架构的角度看一下用户在产品控制台上开启 Indexing service 的能力和内部的流程。

首先，用户在产品控制台开通 Indexing Service 之后，实际索引的 Index metadata 会同步到 Indexing Service，之后写入的流量如 Bulk 和 Doct API 可以通过



节点直接 forward 给 Indexing Service，无需在 Datanode 上构建。索引构建完之后，Indexing Service 可以直接通过跨集群的物理复制直接将索引数据分发到用户集群，从而实现一个秒级的数据时效性。用户也可以在自己的集群内部通过 API 直接查询每个 Index 的订阅状态，并且用索引管理机制管理索引的订阅关系。当然，用户也可以随时取消订阅关系，这时候业务的写入吞吐会 roll back 回自身集群来进行索引构建。

最后总结一下，Indexing Service 是阿里云 ES 基于云上业务实践推出的新产品，基于 Indexing service 的能力，用户可以按需购买，按量付费，大大降低日志和监控等时序场景的机器成本。这个特性将会在 2021 年 4 月份正式上线，也欢迎各位新老用户开通试用。

以上就是云原生 Elasticsearch 驱动新未来的分享的全部内容。



# Elastic Stack 解决方案与行业应用

分享人：Elastic 首席解决方案构架师朱杰

视频地址：<https://developer.aliyun.com/live/246149>

关于 Elastic Stack 如何帮助企业解决问题，本文将通过以下三个部分展开介绍：

- Elastic 三大解决方案
- Elasticsearch 核心优势
- Elasticsearch 行业应用

## 一、Elastic 三大解决方案

经过这么多年的发展，可以用这三个词来概括 Elastic 主要为用户解决的问题：搜索，观测，防护。

基于这三个词，接下来会详细介绍 Elastic 给客户解决的问题。

Elasticsearch  
生态&技术峰会

阿里云

### Elastic三大解决方案



Elastic 企业搜索



Elastic 可观测



Elastic 安全

#### 企业搜索

这个解决方案是 Elastic 历史最为悠久的一个解决方案，因为在很多年前 Elasticsearch 开始变得流行的时候，大家主要是把它用来作为搜索使用的，后来随着应用

的改变，我们又把 Elastic 的一整套工具用作日志、监控等场景。于是就衍生出第二大解决方案。

## Elastic 可观测

这个解决方案除了日志以外还包括了指标，分布式追踪等等，构成了一个全方位的可观测性。

## 安全

整个 Elastic 还被各大安全的解决方案所使用，构成了一个非常庞大的安全的生态系统，所以也把它归纳成 elastic 的安全这一大解决方案。

从 Elastic 自己归纳出来的三大解决方案就可以看到，Elasticsearch 往往在这些方面是被广泛而非常普遍地去使用的。

接下来看一下各个解决方案的具体细节。

### 1. Elastic 企业搜索



## Elastic 企业搜索

Workplace  
Search

App Search

Site Search

主要分为三部分。

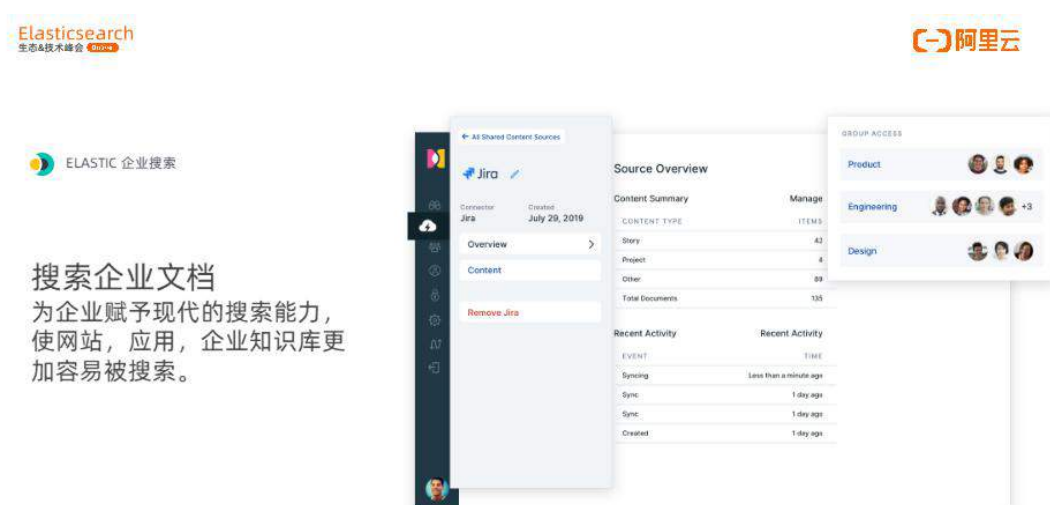
## 1) Workplace Search

这是一个很新的解决方案，对应到企业知识库搜索的领域，它其实是有很多的连接键（connector），可以连接到企业的比如各大网盘，SharePoint，Github 等等数据的来源库，抓取里面企业的数据，然后把它存进 Elasticsearch，去做全文的索引。并且它有一整套完整的 UI 给大家，可以让大家能够非常方便地进行搜索，能够搜索到企业整个范围的所有知识。

## 2) App Search & Site Search

这两个是我们非常熟悉的，比如说我们要开发一个 app 或网站，大家会有非常多全文搜索的需求，现在最为流行的就是使用 Elasticsearch 来为你的 app 和网站提供全文搜索的场景。

下图是 Elastic 企业搜索界面的展示。



可以看到，这里有很多协作的团队，可以基于这样一个庞大知识库帮你去搜到很多项目的文件，很多文档，甚至很多代码。

## 2. Elastic 可观测



## Elastic 可观测

Logs Metrics APM Uptime

这个解决方案最早被大家所使用的主要是做日志。在这个领域，ELK 是非常著名的缩写，指的是 Elasticsearch, logstash, 和 Kibana，它被广泛地使用在各大企业的日志场景。把它叫做可观测，主要是响应现在运维领域非常流行一个趋势。这个趋势就是我们在运维里面都在致力于去结合日志，指标，分布式追踪的数据，还有像 Uptime 这样观测各种 API 是否在线，各种网站是否在线的数据，把他全方位地融合起来，这样就可以对整个企业的 IT 系统从多种角度去看故障和问题。这个在国内外都是一个非常热门的话题，所以把这个解决方案总括地称为 Elastic 的可观测解决方案。

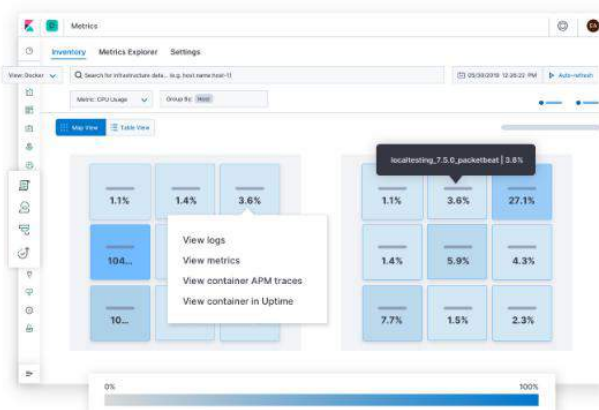
下图是 Elastic 可观测的界面视图。



ELASTIC 可观测

### 对整个环境获得统一观测视图

在一个堆栈内汇集日志，指标和分布式追踪数据，能够以最快速地监控，检测和响应异常



整个界面都体现在整个 Kibana 平台当中的各种 app，它都是由各种 app 所组合起来，构成了这样一个全方位的解决方案。在这些 app 里面就包含了日志的 app，指标的 app，

分布式追踪的 app，还有专门的像 Uptime 这样的应用。这就意味着大家如果使用了 Elastic Stack 一整套工具的话，其实我们已经可以帮你做到对整个环境统一的观测视图。而且大家可以看到，把各种数据融合起来具有非常强大的优势，可以在日志、指标、和 APM 这些数据，在 Kibana 之中非常灵活地进行跳转，不需要去使用很多个这样的系统就能完成对整个环境的观察。

### 3. Elastic 安全



## Elastic 安全

Endpoint SIEM

Elastic 安全已经被很多安全领域的其他厂家普遍使用。最传统的叫作 SIEM，主要是安全事件的管理，往往我们会把企业整个服务器上的，网络的，各种安全设备的等等，包括很多笔记本、台式机上的各种日志，汇总到 Elasticsearch 里面，然后部署很多的安全规则，来对整个企业的安全进行发现，来触发很多安全的事件，进而产生这样的告警。业界这样的产品已经非常之多了，Elastic 作为原厂也会基于整个 Elastic Stack 来提供我们自己的 SIEM 的整套解决方案。

同时，我们和别的厂商的不同还在于，我们还具有一个终端的防御，称之为 Endpoint 防御。这个防御是一个终端的软件，它可以安装在服务器上，笔记本上等等这些终端上面，它不仅仅可以采集这些安全的事件，同时它可以在第一时间做到一个主动的防御，帮你去防御很多恶意的软件，勒索软件，还有很多病毒软件等等，整个采集的数据也会实时地汇总到 Elasticsearch 这边，跟大家的很多网络设备，安全设备的数据聚集起来提供对企业整个安全、全面的可观测性。所以我们的安全解决方案也是非常有新特点的。

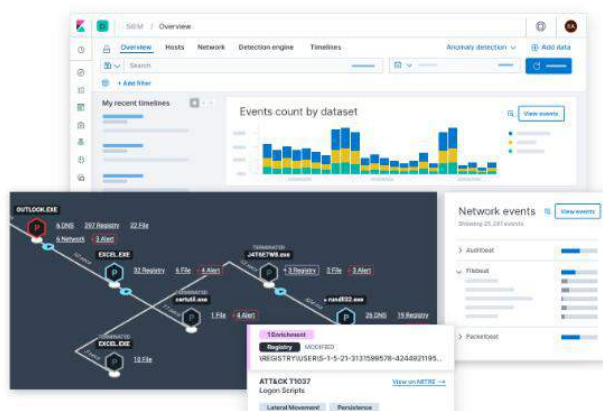
Elasticsearch  
生态&技术峰会

阿里云

ELASTIC 安全

## 融合终端防御和传统 SIEM

Elastic Security 整合了终端安全和 SIEM 赋予对整个基础设施预防、检测和响应能力



上图中可以看到，这是区别于传统 SIEM 的界面。在这里大家可以看到一个终端防御的链路，比如大家点击一个软件，它触发了很多行为，其中有些行为是非常可疑的，这些都可以被这个终端防御的软件识别和采集到，并触发各种各样的告警，第一时间去阻断这种攻击。而且我们的安全管理员也可以基于这些信息进行第一时间的分析和干预，可以把这个终端进行隔离等等操作。

总结一下，Elasticsearch 现在主要为大家解决搜索问题，观测问题，和防御方面的问题，对应了这三大解决方案。

## 二、Elasticsearch 核心优势

### 1. Elastic Stack 坚实的基础

Elasticsearch  
生态&技术峰会

阿里云

## 三大解决方案建立在Stack基础上



Elastic 企业搜索



Elastic 可观测



Elastic 安全



Elastic Stack



这三大解决方案都是建立在 Elastic Stack 上。这个 Stack 已经被大家在社区里广泛使用，它从最底层进行数据的采集，比如 Beats 家族可以采集日志指标等各种数据，还有 ETL 软件，Logstash 主要是在 server 端对数据进行转换、清洗、计算作用。它们采集到的数据会写入到我们最核心的产品 Elasticsearch 里面。

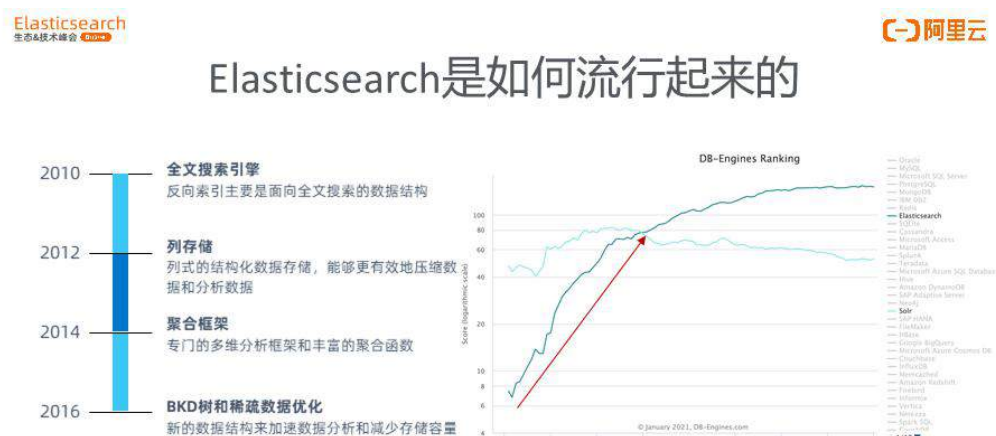
Elasticsearch 是一个大规模分布式的搜索引擎，它不仅仅可以存储大量结构化的数据，同时也会存储很多非结构化的数据，并提供丰富的查询能力，对结构化和非结构化数据进行一致地搜索，然后返回结果。

最上面是报表分析和展示平台 Kibana，同时在这个平台上还有各种各样丰富的解决方案的应用，都是嵌入在 Kibana 这样一个非常灵活和强大的平台之中的，以插件的形式所运行。

所以这一整套就构建了一个非常强大和坚实的基础，使得能够在上面建立起三方面的解决方案，在未来我们还会去夯实每一个解决方案里面丰富的应用，能够给大家带来更好的使用体验。

## 2. Elasticsearch 演进过程

从前面我们看到 Elastic Stack 被应用得非常广泛，在搜索，安全监控领域，安全检测领域等等。这背后离不开这样一个演进的时间轴（如下图）。



2010 年，Elasticsearch 最早被创立的时候，它仅仅是作为一个全文搜索的引擎，主要面向非结构化的文本数据。而几乎对结构化数据的搜索无使用可能，如聚合分析。

2012 年，我们加入了列的存储，有了列存之后，就可以应对比如对很多列的排序，对很多列的聚合计算，这些都需要有比较现代化的面向列的存储才可以比较高效地完成。

2014 年，又专门推出了聚合框架，在以前它主要是面向非结构化的全文性的搜索，但是大家可以想象，其实在现实中很多项目里面，如果要去做很多解决方案一定不能缺少的是聚合的计算，所以在 14 年的时候我们正式推出了这样的聚合框架，使之变成了一个可能，由此能够打开更多的解决方案，应用场景等等。

2016 年，继续对我们结构化数据进行增强，主要是引入了 BKD Tree，能够对数值类型高效地进行存储和搜索，它的加速性能是非常之巨大，同时面向列存进行稀疏数据的优化。这些都是面向于结构化的，所以在演进过程中，Elasticsearch 从一个仅仅面向非结构化的搜索引擎，变成了能够既处理非结构化，又能够处理结构化的全功能的搜索引擎。这个本质上就是 Elasticsearch 为何如此流行的根本原因。

同时从流行度 DB-Engine 的曲线上来看，也可以看出这样的趋势。特别是随着演进时间线，它越来越多地支持结构化数据的搜索和统计之后，流行的程度非常快地增长，到了 16 年我们已经构建得比较完备的时候，就彻底超越了像 Solar 这样的老牌系统。后来 Elasticsearch 基本上就占据了搜索引擎的一个主导地位。

### 3. 核心优势总结

总结一下，Elasticsearch 之所以大受欢迎，主要的核心优势就是这三个词语：speed, scale 和 relevance，速度，扩展性和相关性。所以本质上 Elasticsearch 是一个搜索公司，因此我们致力于提供非常快速的搜索体验，并且能够返回最符合搜索者要求的这些搜索结果，但更加关键的是我们能够把这种搜索能力扩展到海量的数据，这就是 Elastic 的核心优势。

## 三、Elasticsearch 行业应用

接下来以一个非常现实的搜索问题入手，来看看 Elasticsearch 强大的查询和聚合能力。比如场景是搜索一个餐厅，往往条件首先是这个餐厅的评分，假定它的评分是大于三分的，这个属于结构化搜索。还需要包含一些特色菜肴，比如搜索羊肉这道菜，属于全文搜索类型，非结构化自动搜索。第三个条件是要搜索距离 1km 以内的符合条件的餐厅，属于地

理位置的经纬度搜索。这个搜索必须得把这三部分不同类型的搜索，用 and 的连接符连接起来，并返回这个结果。Elasticsearch 就可以提供这样强大的搜索能力，帮你完成这样复杂的搜索要求。搜索完成之后，它会返回一个命中的列表，还要针对这个返回的具体列表做一些聚合统计。会统计各种菜的菜系分别包含了哪些餐厅，这个就是非常典型的聚合计算的场景，所以 Elasticsearch 就可以赋予大家这样丰富的查询能力，并且赋予强大的聚合查询能力。

正是 ES 强大的搜索和聚合能力，才造就了现在应用场景的空前繁荣。



## 应用场景空前繁荣



这个丰富的词云里面包含了各种行业，其中有最为普遍的金融行业，还有互联网行业，很多是我们每天的生活都是离不开的，比如点外卖，购物，有订单的搜索，背后其实都含有 Elasticsearch，还有很多快递运单的查询等等，所以大家可以看到 Elasticsearch 能应用的场景是非常丰富的。

下面就以一些行业为代表，分别去看一下 Elasticsearch 在这些非常有代表性的行业里面的应用。

### 1. 金融行业

#### 1) 业务搜索

从业务层面来讲，比如征信系统，风控系统等等都是可以用 Elasticsearch 的。背后的逻辑主要还是一个标签系统的逻辑，对于贷款人员等等会调用很多的算法，基于历史数据

来产生很多用户画像的标签，基于这些标签就可以设定很多规则的引擎，然后通过 Elasticsearch 强大和快速的搜索能力，能够快速地对这些用户来产生画像和评分，进而决定他是否符合贷款条件等等。还有像金融产品的搜索推荐，本质上也是建立在用户的画像基础上。在保险行业，各大保险公司经常用的是保单查询这一业务场景，因为保单既包含结构化的数据，又包含非结构化的数据，所以要合起来搜索条件还是非常苛刻的。广告系统，很多金融的产品要去推荐给各种客户，同样需要依赖于所产生的用户画像和用户标签，进而去筛选出这些对应的用户定向推送。这些都可以用 Elasticsearch 强大的搜索能力来进行。

## 2) 数据分析

数据分析层面，往往可以做一些交易的分析和统计，代表着我们会把很多交易的记录导入到 Elasticsearch 里面，还包含了像用户分析，数据分析和统计以及商户分析等等。

## 3) 系统运维安全

大量金融机构也在用 Elastic Stack 整个一套工具来进行日志集中化的管理。在日志里面还可能包含对于整个交易系统，全链路日志的串联，能够把分布在各个交易系统的并通过 Transaction ID 串联到一起，来进行故障的诊断和客服的应对。客服也会有日志的查询，对应着我们的开发团队会有故障的定位等等。同样，整个金融系统也大量地会把一些安全的数据写到 Elasticsearch 里面，然后来部署规则，进行整个安全系统的检测。

# 2. 电商行业

## 1) 业务搜索

首先，海量的商品需要进行数据的搜索，有很多个字段。电商所产生的订单数据量也比较大，同样它也需要丰富的搜索条件，这些都可以用 Elasticsearch 来更高效地完成多条件的，甚至是全条件的搜索。基于用户标签，它也可以做很多用户特征的搜索，能够定制化地为用户提供产品推荐。同样的，在电商领域也会有很多非结构化的记录产生，比如客服对话，用户评论等等，都会使用到 Elasticsearch 来进行搜索和定位。像物流运单的查询，更多的是结构化的查询的场景，它会把压力从数据库转移到 Elasticsearch 里来进行查询。

## 2) 数据分析

在这一个领域，它可以做交易数据的分析，勾勒用户的画像，分析商户的行为，店铺的这些数据行为等。

### 3) 安全运维

同样的，各种大量的日志也会放在 Elastic Stack 当中，同样应对着客服，IT 运维来进行。还可以用作 APM 分布式追踪，和系统安全的检测。

## 3. 在线教育

### 1) 业务搜索

在业务搜索里有教育领域非常典型的在线课程的搜索。还有比如试题的搜索，试题关联会有向量的匹配，相似的试题往往会编码成各种向量，然后对相似的向量进行搜索。还会有教案的搜索，是包含丰富文本的。同样也有订单、客服的查询，这些都是有共性的，所以大家可以联想到这些共性的东西就是可以用 Elasticsearch 做。

### 2) 数据分析

会有很多在线直播的分析、监控等，因为在线的课程会有很多进入直播间，退出直播间，还有学生的反应等等，这些都可以用 Elasticsearch 来对整个直播进程进行分析。同样，根据画像系统也可以对教师、学生进行分析。

### 3) 系统运维安全

包括在线课程的全链路的观测，全的日志，还有 API 的质量等等，都可以通过 Elasticsearch 来对整个运维提供更好的数据支撑。

## 4. 游戏行业

### 1) 业务搜索

对游戏行业论坛是至关重要的，ES 会支持游戏论坛的搜索。还有道具，商品等等，也离不开商品的搜索。还有和电商、在线教育都非常类似的订单和客服支持的信息查询。

### 2) 数据分析

游戏的运营会涉及到很多用户的购买，购买的喜好，可以对他进行分析后进行推荐，来产生更好的效益。各种道具的购买，推荐等等，都可以用到 Elasticsearch 的能力，帮助大家提高运营的水平 and 收入。

### 3) 系统运维安全

同样包括系统和应用日志的分析。还有安全，在游戏行业是非常重要的，都可以通过 Elasticsearch 帮助大家进行系统运维的安全和监控。

## 5. 汽车制造业

### 1) 业务搜索

有些汽车制造会使用一些配件的搜索，还有一些汽车的 4S 管理也会用到 Elasticsearch 来进行业务数据的搜索。还有和车联网相关的创新应用，现在在很多汽车上面都会有很多车机系统，会有这方面业务的数据搜索。

### 2) 数据分析

刚刚提到的车联网业务，会产生很多用户数据，通过车联网发送到云端，而在云端很多都是放在 Elasticsearch 里面来对车联网的数据进行多维的搜索和分析。所以在数据分析会有车联网用户的体验分析，设备的画像，以及现在比较新颖的自动驾驶，它会采集大量数据，比如路测的各种数据，很多汽车单位会把这些数据产生的原标签放到 Elasticsearch，来进行搜索。与之关联的就是 IOT，设备生产线上的很多数据也可以用到 Elasticsearch

### 3) 系统运维安全

同样的，是对整个系统的日志，还有车联网微服务的分布式追踪等等，这些都可以用到 Elasticsearch。

以上就是 Elastic Stack 解决方案与行业应用的全分享内容。



# 阿里云 Elasticsearch 云原生内核

分享人：阿里巴巴集团技术专家魏子珺

视频地址：<https://developer.aliyun.com/live/246150>

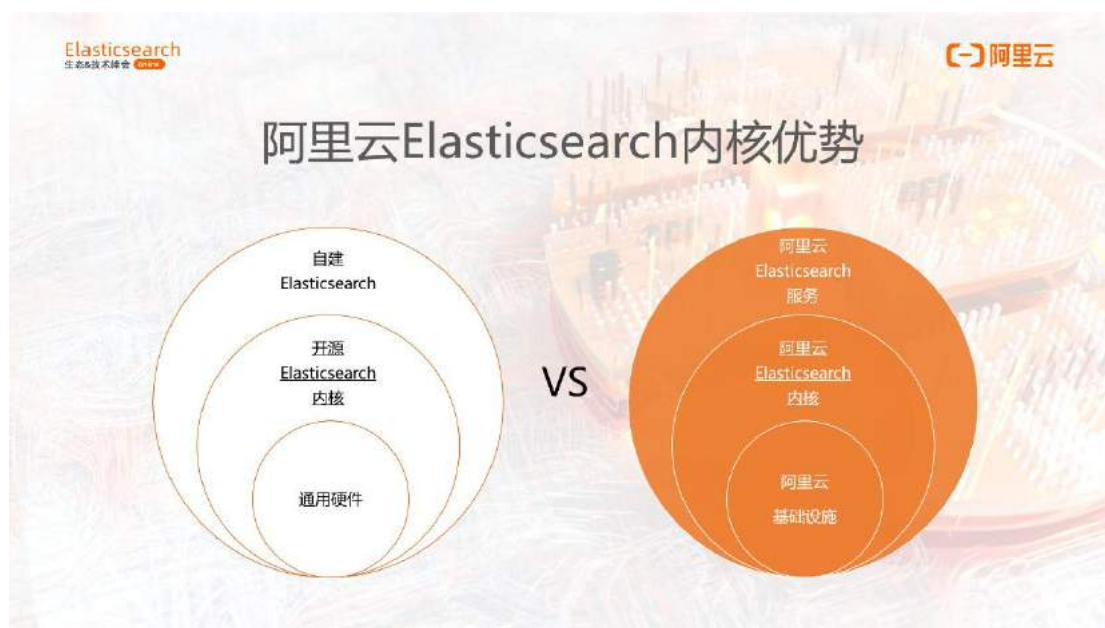
关于阿里云 Elasticsearch 云原生内核，本文将通过三个部分展开介绍：

- 阿里云 Elasticsearch 内核概览
- 云原生 Elasticsearch 定义
- 阿里云原生 Elasticsearch 实践

## 一、阿里云 Elasticsearch 内核概览

### 1. 阿里云 ES 的优势

下面这个对比图可以很好地说明阿里云 ES 相比开源 ES 的优势。



先看内圈，开源 ES 针对通用硬件设施，而阿里云 ES 内核则是针对阿里云基础设施深度定制的内核，可以最大地发挥阿里云基础设施的性能，以及成本方面的优势。然后，看最外圈，开源 ES 内核为了适应 ES 丰富的使用场景包括搜索，可观察性等，无法做到功能和性能兼顾，而阿里云 ES 内核依托于阿里云 ES 的服务可以做很多场景化的优化和功能增强，在搜索和观察性等方面会比自建的 ES 更有优势。内核在中圈，向下运行在阿里云基础设施，向上依托于阿里云 ES 服务。可以看到，阿里云 ES 内核相比开源 ES 自建集群，不论在成本、性能、稳定性和功能上都会更具优势。

## 2. 基于用户需求的内核建设

用户对阿里云 ES 内核的需求，主要是以下三个方面：

### 1) 简单

存储容量能够不断扩充，计算有足够的弹性，用户不需要操心资源的问题。

### 2) 好用

开箱即用，不用进行一系列复杂的部署和配置，直接根据场景提供最优的配置即可，还要有丰富的检索功能可以使用。

### 3) 性价比

阿里云 ES 做到价格足够低，性能足够好，足够稳定。

结合上述的需求，我们从下面四个方面展开内核建设。



### 1) 成本节约

第一，我们提供计算存储分离的增强版 ES，成本上节省了副本的开销，保证足够的弹性，可按需使用。第二，支持冷热分离，成本更低地存储介质。第三，Indexing Service，这是我们全新发布的产品，对写多读少的场景有很大的成本节约。第四，索引数据压缩，我们新增的压缩算法可以比默认的压缩方式提升 45% 的压缩率。

### 2) 性能优化

第一，我们研发的 ElasticBuild 相比在线的写入能有三倍的性能提升。第二，我们还研发了物理复制功能，从最早支持计算存储分离到现在的支持通用版的 ES，物理复制通过 segment 同步而不是 request 同步的方式，减少了副本的写入开销，所以有一个副本的情况下，写入性能能有 45% 左右的性能提升，副本越多，提升越明显。第三，bulk 聚合插件，在协调节点聚合下载的数据，降低分布式的长尾效应，在写入吞吐高、分辨数多的场景写入吞吐能有 20% 以上的性能提升。第四，时序查询优化，针对 range 查询，可以直接跳过不在 range 范围内的 segment，结合时序策略可以获得更好的查询性能提升。

### 3) 稳定性提升

第一，我们研发了集群限流插件，能够实现索引，节点，集群级别的读写限流，在关键时刻对指定的索引降级，将流量控制在合适的范围内。第二，慢查询隔离池的特性，它避免了异常查询消耗资源过高导致集群异常的问题。第三，协调节点流控插件，它集成了淘宝搜索核心的流控能力，针对分布式环境中偶发节点异常导致的查询抖动，能够做到秒级切流，最大程度降低业务抖动概率，保证业务平稳地运行。第四，monitor 插件，它采集了集群多维度的指标，可以提供全方位的监控。

### 4) 功能增强

第一，向量检索插件，是基于阿里巴巴达摩院 Proxima 向量检索库实现，能够帮助用户快速实现图像搜索、视频指纹采样、人脸识别、语音识别等等场景的需求。第二，阿里 NLP 的分词插件，它是基于阿里巴巴达摩院提供的阿里 NLP 的分词技术，支持阿里内部包括淘宝搜索、优酷、口碑等业务，提供了近 1G 的海量词库。第三，OSS 的 Snapshot 插件，它支持使用阿里云 OSS 的对象存储来保存 ES 的 Snapshot。第四，场景化的推荐模板，可以针对不同的业务场景提供成本、性能的优化。

以上的这些特性都能在我们阿里云 ES 官方文档中看到，欢迎大家使用。

## 二、云原生 ES 的定义

### 1. 云原生 Elasticsearch 如何定义

首先看一下什么是云原生。阿里巴巴云原生公众号前段时间推出了一篇文章《什么是真正的云原生》，总结了云原生的定义：第一，弹性、API 自动化部署和运维；第二，服务化云原生产品；第三，因云而生的软硬一体化架构。



上图是云原生架构的白皮书封面，这是由阿里云二十位云原生技术专家共同编写，已经正式对外发布，欢迎大家阅读。

那么，什么是云原生 ES？

- ES 的云服务，开箱即用，能用 API 自动化部署和运维
- 计算存储分离，弹性可伸缩
- 能充分利用云基础设施，网络、存储和算力等
- 以上三点就能对应最开始提到的三个圈： 服务，内核，和基础设施，这样才是云原生 ES。

### 2. 云原生 Elasticsearch 如何设计



第一，它必须是计算存储分离的架构，这样才能提供更加弹性的计算能力和无限的存储空间。第二，可以支持冷热分离，冷热的节点都要是计算存储分离的架构。冷节点使用高性价比的对象存储，相比热节点可以有 90% 的成本节约。第三，Serverless 的用户真正关心的是索引的使用，而不是 ES 集群的维护。Serverless 让用户的关注点从集群的维度可以下沉到索引维度。

### 3. 云原生 Elasticsearch 内核挑战

实现这样的云原生内核，挑战是非常大的，主要的挑战分为下面三个方面：

#### 1) 热节点的分布式文件系统

第一，分布式文件系统自身的稳定性保证，ES 对 Latency 非常敏感，它提供性能和稳定性与本地盘相当的分布式文件系统，这个挑战本身就非常大。第二，ES 在实现一写多读时，如何防止出现多写的情况。ES 数据是在内存，读写需要的内存状态数据、数据是如何保持一致性的，这些都是很大的挑战。

#### 2) 冷节点的对象存储

对象存储提供的是 HTTP 接口，所以需要去适配。另外，对象存储的单次 IO Latency 非常高，所以只有在冷节点相对 Latency 不敏感的场景才有机会使用。如何解决 Latency 最高的问题也是很有挑战。最后是对象存储无法使用到操作系统的 pagecache 和预读能力，所以要用对象存储，这些能力必须在 ES 侧实现。



### 3) Serverless

最难解决的就是多租户的共享和隔离的平衡问题，如果不同索引直接产生相互的影响，在云上是不可接受的。如果不共享，就意味着资源无法充分利用。如何平衡共享和隔离的问题，这是 Serverless 最大的挑战。第二是体验，基于索引的使用如何提供和云原生 ES 一样的体验也是需要考虑的问题。第三是资源监控，如何评估索引的使用资源也是一个挑战。

## 三、阿里云原生 Elasticsearch 实践

### 1. 计算存储分离



计算存储分离核心的诉求是弹性，它不只是像云原生 ES 那样支持动态的添加节点、自动 Shard 搬迁，而是彻底的弹性。对于 ES 来说，它的核心诉求是两点：Shard 秒级搬迁和 Replication 秒级增加。这样才能解决热点的问题，和高峰快速的动态扩容的问题。如果扩缩容还要迁移 Shard 的数据，弹性是不够的。彻底的弹性一定是 Shard 搬迁，Replication 扩充，数据是不动的，只是调整 DataNode 对 Shard 的映射。要实现这样的弹性，就必须做到计算存储分离。

阿里云对 ES 存储分离内核的实现如下：

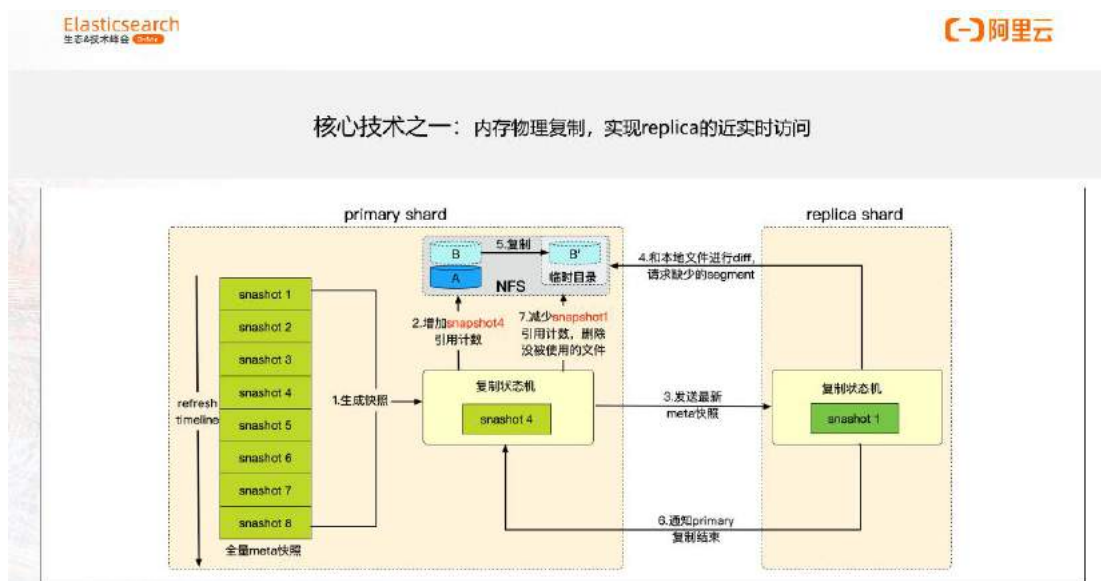
- 数据存储在分布式文件系统上，由分布式文件系统保证数据的可靠性。
- 同一个 Shard 的多个副本数据只保存一份。
- 一写多读的场景，这样就不再依赖于 ES 自身的 replication，可以减少写入的开销。



- 索引扩 Shard，无需复制数据，秒级增加只读 Shard。
- Shard 搬迁无需迁移数据，秒级切换 DataNode。

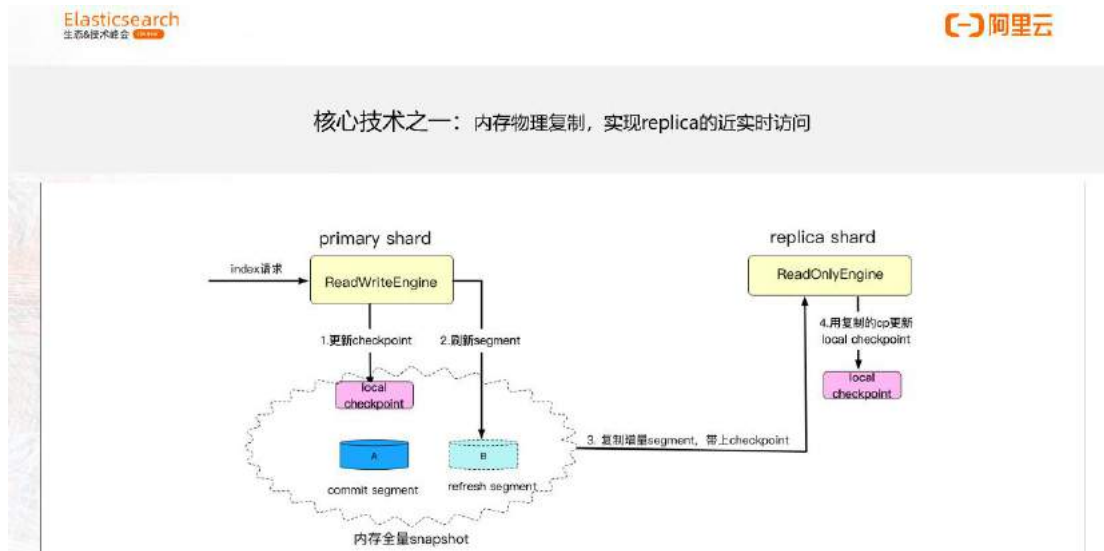
### 核心技术之一：内存物理复制，实现 replica 的近实时访问

Segment 同步的实现细节：



图中描述的是 ES 物理复制的状态机，核心是为了解决 segment 同步乱序的问题。通用的物理复制功能也是一样的实现，主要区别在于计算存储分离只需要复制实时生成的 segment，对于后续产生的 segment，强制提交 commit，确保 segment 落盘，来防止大的 segment 进行复制。而通用的物理复制，外界的 segment 也是需要复制的，这种 segment 往往会比较大。所以这里有一个关键的优化，为了防止大 segment 复制导致的主从可见性差距过大，主 shard 在从 shard 复制完成后才会打开最新的 segment。

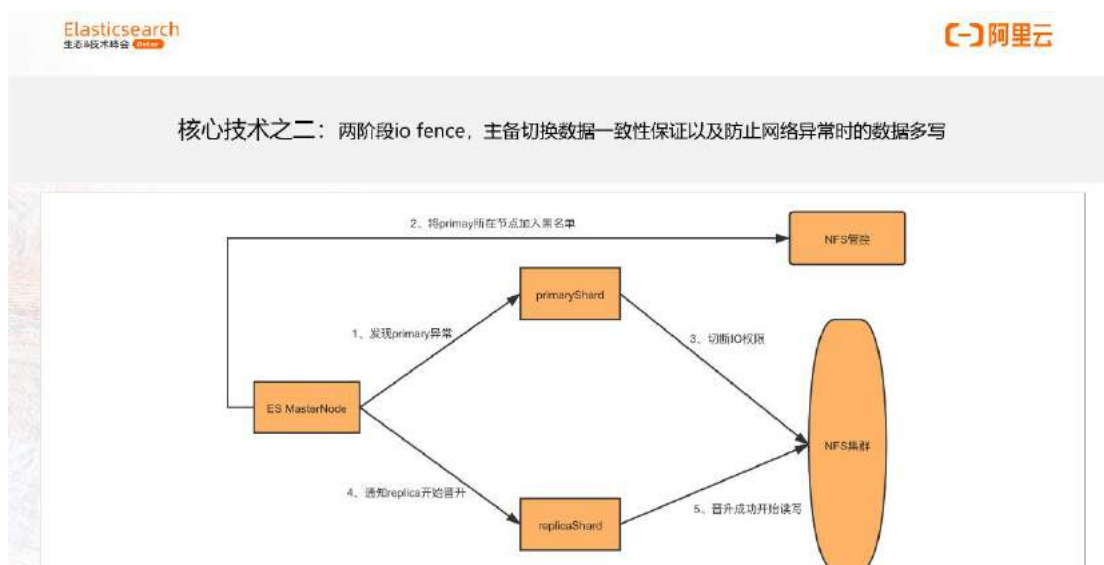
下图介绍了物理复制保证数据一致性的方式。



核心是保证 checkpoint 的一致性，通过将主 shard 的 checkpoint 同步到从 shard 来实现。结合这张图可以看下流程，当数据写进来的时候，主 shard 会更新 checkpoint，在第二步刷新 segment 时，第三步将 segment 复制到从 shard 时，会带上 checkpoint，第四步从 shard 会用这个 checkpoint 更新自己的 local checkpoint 来保证主从 shard 使用了相同的 checkpoint，这样就实现了数据一致性的保证。

## 核心技术之二：两阶段 IO fence

核心要解决的问题是防止多写。通过分布式文件系统的管控侧将异常节点加入黑名单，直接从根本上防止了异常节点的显露。



上图展示了整体的流程，在主 Shard 节点异常的时候，MasterNode 首先发现主 Shard 的异常，然后将主 Shard 所在的节点加入黑名单。第三步，这个节点切断了 IO 的权限，彻底失去了写的能力。第四步，master 通知从 Shard 晋升成主 Shard。第五步，从 Shard 晋升成主 Shard 后，就开始正常地读写数据。

我们的计算存储分离的架构通过阿里云增强版进行售卖。计算存储分离除了弹性的特点外，由于一写多读的特点，在性能、成本上都有显著的提升。我们测试了线上阿里云增强版 ES 和原生 ES 在同样规格配置的性能对比情况，从表格的最右一列红色的标识可以看到，不论在 translog 同步还是异步的场景，一个副本的情况下，性能都有超过百分之百的提升，副本越多，性能提升越明显。



总结一下计算存储分离的特点：首先它是秒级弹性扩缩容；第二，由于不写副本，所以写入性能能有 100% 的提升；第三，由于多个副本存储一份数据，所以存储成本呈倍数降低。

### 计算存储分离——热节点

想要使用我们计算存储分离的 ES 集群，可以选择图中所示的日志增强版，欢迎大家使用。



计算存储分离——冷节点

热节点通过分布式文件系统实现了计算存储的分离,冷节点也需要实现计算存储分离才能实现弹性。冷节点这部分我们还在研发阶段, 所以这次分享给大家的是一些思考的内容。



冷节点的成本是第一要素, 所以对象存储成了首选。对象存储相比分布式文件系统和块存储等特点非常鲜明。劣势, 大都在挑战中提及到, 这些劣势相比其他存储, 无论从易用性和性能上都无法跟分布式文件系统和块存储相比, 所以这些热节点很难直接使用对象存储。但是冷节点不同, 冷节点核心考虑的是成本, 因此它也有一些优势。它的成本比 SSD 云盘

便宜近 90%，可以真正的按需使用，不用预先准备存储空间。另外可以提供 12 个 9 的可靠性，所以也可以不用存储副本，这又是一半以上的成本节约。基于这些优势，对象存储成了最好的选择。

如何最大减少它的劣势带来的影响，这要从 ES 的特点说起。ES 在可观察性、安全的方向上，冷热数据明显，日志长期存在 SSD 上成本过高，所以可以考虑冷热架构。第二，ES 在 search 的时候很消耗 CPU，因此可以利用计算时异步地扒取对象存储的数据，减少 IO 等待的时间。第三点是冷数据基本上无写入，所以对写入性能要求也不高。以上的三点就是 ES 冷数据使用对象存储的原因。

## Indexing Service

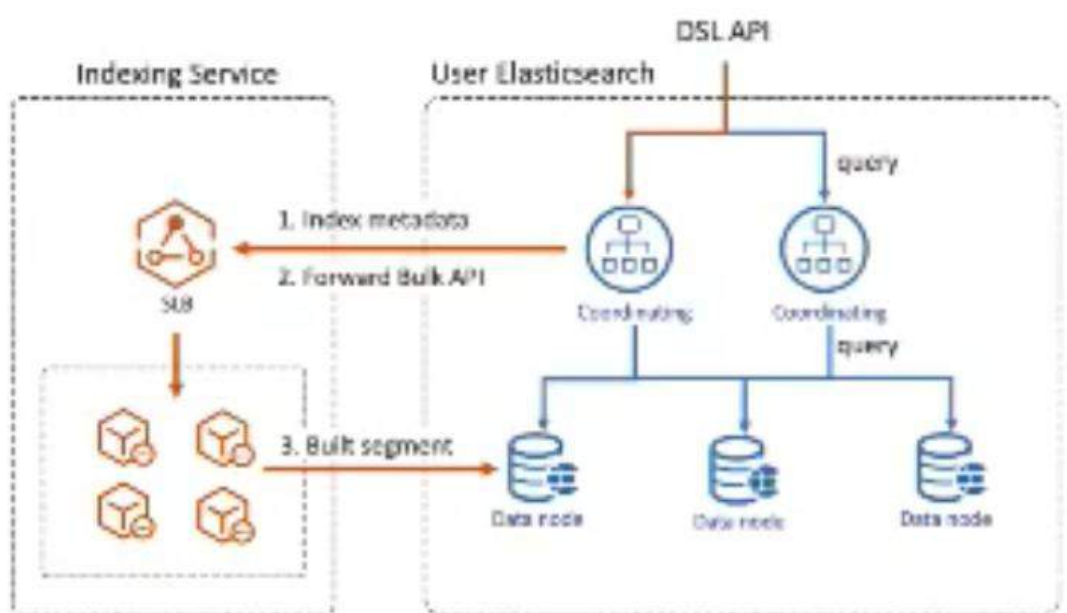
Indexing service 是我们即将重量级发布的新产品，这是我们在 serverless 尝试的一个产品，是针对写入方面的性能优化。相比查询的多样性，写入会相对简洁明了。Indexing service，从功能方面，提供了写入托管服务，满足高并发持续数据写入，降低了业务集群的 CPU 开销，它适用在日志、监控、APM 等时序场景，它解决的最大痛点是写多读少，而且很多时序场景下写远多于读，业务需要消耗大量的节点资源来满足写入。Indexing service 可以极大的降低这部分场景的成本开销。

Indexing service 有以下三个方面的特点：

- 完全兼容原生的 API
- 按量付费，按写入吞吐和 QPS 实际需求付费
- 写入能力可秒级快速弹性扩缩

下图展示了 Indexing service 是如何实现的：





第一，请求转发，请求发到用户 ES 集群，用户使用云原生 API 操作 ES，ES 内核会将开启托管的索引写入请求，转发到 Indexing Service。这里可以展开再缩小，对于不再写入的索引，用户就可以取消协助托管，释放存储成本。Indexing service 结合 Data stream 和 over 功能可以有非常好的用户体验，因为新生成了索引后，老索引就不再写入。我们在内核上做了优化，在生成新索引的时候就会自动取消托管。

第二步，在写入 Indexing service 后，内部会经过分布式的 QoS 模块，进行写入的流量控制，来阻止资源的过度消耗。

第三步，跨集群的物理复制，Indexing Service 构建的索引是通过物理复制到用户集群的。

最后是 Indexing Service 内部会持续地运行原数据同步的 task，实时地同步用户集群托管的索引 metadata。

Indexing Service 将于 2021 年 2 月全新上线,实现 ES 在时序日志场景的降本增效。Index server 可以解决时序日志数据高并发写入瓶颈，优化集群写入计算资源成本，降低运维的复杂度。Indexing Service 无论从写入性能，成本节约，还是弹性伸缩的能力方面都能带来不一样的体验，大家可以敬请期待。



# 如何规划和执行威胁狩猎

分享人：Elastic 社区布道师刘征

视频地址：<https://developer.aliyun.com/live/246151>

关于如何规划和执行威胁狩猎，本文将通过三个部分展开介绍：

- 流行的攻击生命周期框架
- 规划可重复的威胁狩猎流程
- 安全威胁检测规则的管理

## 一、流行的攻击生命周期框架

首先看一下目前企业可能面临的安全威胁和一些风险。目前，不管企业规模大小，几乎所有的员工都使用着各种各样的 IT 基础设施和服务，因此我们的攻击面就充满了各种各样的安全盲点。每一个服务，甚至每一个人都可能是被攻击的目标和被黑客袭击的对象。

Elasticsearch  
搜索即服务

阿里云

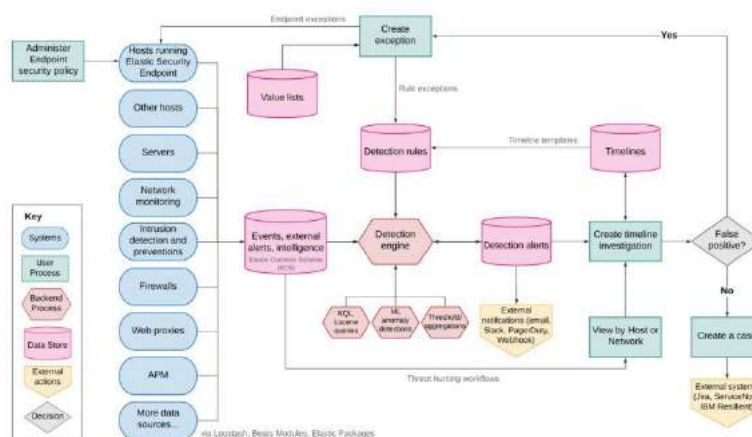
① 攻击面充满了盲点

② 每个人都是目标

③ 安全分析师不堪重负

我们组织里面的安全分析师也早已是四面楚歌，不堪重负。在 Elastic Stack 这样一个技术创新发展的演进过程当中，Elastic Stack 逐渐形成了一个日臻完备的威胁狩猎平台。

## Elastic 日臻完备的威胁狩猎平台



在当前最新版的 Elastic Stack 技术栈中，我们内置了在威胁狩猎方面非常重要的一个核心组件，这个组件包括了检测规则和检测引擎。检测规则和检测引擎将利用大量的 ESC 兼容的各种安全的告警和事件，这些安全告警事件都是通过我们现有的 Beats，Logstash 和 elastic endpoint 模块采集和集成过来的。除了具有检测规则和检测引擎之外，我们还在右侧这边集成了很多自动化的工作流，包括检测的告警，基于时间线的案例研究，外部的安全事故管理，和流程平台的一些对接。在 Elastic Stack 技术堆栈中，我们可以使用它来做一些安全事件管理，并基于安全事件管理的数据基础做比较可行和完备的威胁狩猎安全管理工作。

## 二、规划可重复的威胁狩猎流程

### 1. 【安全威胁建模步骤】

如果我们想做威胁狩猎，就需要对安全威胁进行建模。在一个纷繁复杂的 IT 环境中，我们首先要有威胁的假设，然后基于这些假设做管理。有如下四个步骤：

- 谁是你的敌人？
- 敌人的动机？
- 敌人的目标？
- 攻击成功后会有什么影响和损失？

基于这个初步思索,我们会再用到一些比较缜密的业内流行的安全威胁攻击的生命周期的框架。





## MITRE ATT&CK - 企业矩阵

框架介绍

	Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Tactics 战术	Drive-by Compromise	Remote Desktop	Task Scheduler	Access Tokens	Account Manipulation	Account Discovery	Applet Script	Auto Capture	Automated Tasking	Command and Control	Command and Control
	External Remote Desktop	CVSS	Accessibility Features	Accessibility Features	BitLocker	Dark History	Application Windows	Application Deployment	Automated Collection	Data Compression	Communication Through Remote Media
	Malware Addition	Command-line Interface	Applet DLLs	Applet DLLs	Binary Packing	Dark Pages	Browser Bookmarks	Distributed Component	Clipboard Data	Data Encrypted	Connection Proxy
	Control Panel Items	Applet DLLs	Applet DLLs	Process User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data Staged	Data Transfer (Over Link)	Custom Command and Control Protocol	
	Dynamic Data Storage	Application Shimming	Application Shimming	CVSS	Credentials in Files	Network Service Scanning	Login Scripts	Data from Information Repositories	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol	
	Stealthful Link	Authentication Package	System User Account Control	Clear Command History	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Local System	Exfiltration Over Command and Control Channel	Data Encrypted	
	Spreading via Remote	Execution through Remote	BITS Jobs	DLL Search Order Hijacking	Dark Signing	Exploitation for Credentials	Pass the Ticket	Data from Network Shared Drive	Exfiltration Over Other Network Medium	Data Encrypted	
	Supply Chain Compromise	Exploitation for Client Execution	Bootkit	Dynamic Hijacking	Component Firmware	Process Automation	Remote Device Discovery	Data from Remote Media	Exfiltration Over Physical Medium	Domain Fronting	
	Trusted Relationship	Graphical User Interface	Browser Extensions	Exploitation for Privilege Escalation	Component Object Model Hijacking	Hooking	Permissions Groups Discovery	Remote File Copy	Exfiltration Over Physical Medium	Exfiltration Over Physical Medium	
	Valid Accounts	WeakAuth	Change Default File Association	Kernel Driver Memory Injection	Control Panel Items	Host Capture	Process Discovery	Remote Services	Exfiltration Over Physical Medium	Exfiltration Over Physical Medium	
Techniques 技术	Local Job Scheduling	Device Account	Image File Execution Options Hijacking	DLL Search Order Hijacking	Dark Signing	Host Capture	Process Discovery	Remote Services	Exfiltration Over Physical Medium	Exfiltration Over Physical Medium	
	Malware	DLL Search Order Hijacking	Launch Hijacking	Dark Signing	Dark Signing	Host Capture	Process Discovery	Remote Services	Exfiltration Over Physical Medium	Exfiltration Over Physical Medium	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	
	Powercat	Dynamic Hijacking	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	Powercat	

我们推荐使用 Mitre ATT&CK 的企业矩阵, Mitre ATT&CK 的企业矩阵, 将所有的安全威胁攻击都通过战术, 技术和流程等方式有条理地进行梳理。比如在这个表格中, 表头包括初始的入侵, 入侵后的执行, 提权操作。在你的网络中发现, 网络中信息往外的渗透, 并执行一些命令和控制。上面都是一些战术, 这些战术可以分阶段地组合成对我们企业 IT 资产和服务人员不同的攻击行为。在每一个战术中, 也就是每一列, 都分别列出了不同使用的技术, 这些攻击技术中都是使用这种字母排序的形式给我们做梳理和展示。所以当企业在做威胁狩猎的时候, 我们就非常推荐参考 Mitre ATT&CK 的企业矩阵模型。

## 2. 【典型事例】

我们从中找出一个非常典型的事例, 来看一下这个矩阵模型告诉了我们什么。

Elasticsearch 生态技术峰会

Home > Techniques > Enterprise > Phishing > Spearphishing Attachment

## Phishing: Spearphishing Attachment

Other sub-techniques of Phishing (3)

ID	Name
T1566.001	Spearphishing Attachment
T1566.002	Spearphishing Link
T1566.003	Spearphishing via Service

Adversaries may send spearphishing emails with a malicious attachment in an attempt to gain access to victim systems. Spearphishing attachment is a specific variant of spearphishing. Spearphishing attachment is different from other forms of spearphishing in that it employs the use of malware attached to an email. All forms of spearphishing are electronically delivered social engineering targeted at a specific individual, company, or industry. In this scenario, adversaries attach a file to the spearphishing email and usually rely upon user execution to gain resolution.

There are many options for the attachment such as Microsoft Office documents, executables, PDFs, or archived files. Upon opening the attachment (and potentially clicking past protections), the adversary's payload exploits a vulnerability or directly executes on the user's system. The text of the spearphishing email usually tries to give a plausible reason why the file should be opened, and may explain how to bypass system protections in order to do so. The email may also contain instructions on how to decrypt an attachment, such as a zip file password. In order to evade email boundary defenses, adversaries frequently manipulate file extensions and icons in order to make attachments appear to be document files, or files exploring one application appear to be a file for a different one.

Procedure Examples

Name	Description
admin@308	admin@308 has sent emails with malicious Microsoft Office documents attached. <sup>[1]</sup>
APT-C-36	APT-C-36 has used spearphishing emails with password protected RAR attachment to avoid being detected by the email gateway. <sup>[2]</sup>
APT1	APT1 has sent spearphishing emails containing malicious attachments. <sup>[3]</sup>

https://attack.mitre.org/techniques/T1566/001/

Techniques 技术

Tactics 战术

检测数据源

参考过程

下面有一个网址，这个网址中非常详细地叙述了鱼叉式钓鱼攻击这样一个攻击的技术。在这个网页上我们可以看见这个攻击的技术还可以和其他三个技术相关联，比如这个网页是基于邮件附件的鱼叉式钓鱼攻击，这个鱼叉式钓鱼攻击还可以和邮件的链接相关，还可以跟钓鱼的服务相关。我们可以看到在这样一个非常详尽的说明当中，有关于这个战术的所处阶段，也就是 Mitre ATT&CK 的企业矩阵的表头部分，每一种特定的攻击有一个具体的 ID，这些信息将让我们的安全管理人员互相之间的沟通更加专业一致。在这个案例中，我们可能就以安全威胁 ID t1566.01 这样一个 ID 来做文档中沟通的一个核心信息。

Elasticsearch 生态技术峰会

Home > Techniques > Enterprise > Phishing > Spearphishing Attachment

## Mitigations

Mitigation	Description
Antivirus/Antimalware	Anti-virus can also automatically quarantine suspicious files.
Network Intrusion Prevention	Network intrusion prevention systems and systems designed to scan and remove malicious email attachments can be used to block activity.
Restrict Web-Based Content	Block unknown or unused attachments by default that should not be transmitted over email as a best practice to prevent some vectors, such as .scr, .exe, .pif, .cpl, etc. Some email scanning devices can open and analyze compressed and encrypted formats, such as zip and rar that may be used to conceal malicious attachments.
User Training	Users can be trained to identify social engineering techniques and spearphishing emails.

## Detection

Network intrusion detection systems and email gateways can be used to detect spearphishing with malicious attachments in transit. Detonation chambers may also be used to identify malicious attachments. Solutions can be signature and behavior based, but adversaries may construct attachments in a way to avoid these systems.

Anti-virus can potentially detect malicious documents and attachments as they're scanned to be stored on the email server or on the user's computer. Endpoint sensing or network sensing can potentially detect malicious events once the attachment is opened (such as a Microsoft Word document or PDF reaching out to the internet or spawning Powershell.exe) for techniques such as Exploitation for Client Execution or usage of malicious scripts.

https://attack.mitre.org/techniques/T1566/001/


预防缓解降低风险防御技术

检测方式防御策略

如果我们想应对这样的安全威胁，我们也要去利用到现有的一些检测的数据源。安全威胁攻击的流程大致是像上图表格中的步骤，Mitre ATT&CK 中不仅描述了某一个特定的安全攻击行为的特征，而且还提出了相应的防御技术，以及预防缓解的方法。比如在这个鱼叉式攻击当中，我们可以使用防病毒软件，防恶意软件，IDS，网络的防入侵检测系统等帮我们做抵御。另外，我们检测这种安全威胁的发生也有一些特定的策略，比如这里使用网络的入侵检测系统，IDS，或者是防病毒的一些实践，报告检测，和相应的威胁风险的迹象。在 Mitre ATT&CK 的官网上一共描述了大概有两百多种典型的攻击行为的特征，有些是像这种鱼叉式钓鱼攻击比较详细的描述，有些就描述地比较概要。因此 Mitre ATT&CK 是一个非常可信的参考框架，让我们来对威胁狩猎的这种工作进行组织，可以作为一个参考的源头。

### 3. 【经典案例分析】

再举一个更详细的案例，刚才讲到的是鱼叉式钓鱼攻击，我们用一个可能更普遍的一种攻击为案例来看一下我们如何在 Elastic Stack 当中去实施 bits Jobs 的安全风险的狩猎过程。

  
生态&技术峰会



## 狩猎假设案例

BITS Jobs

#### 识别 BITS Jobs 的 TTP

- Tactic 战术: Persistence 存在 (TA0003)
- Technique 技术: BITS Jobs (T11197)
- Procedure 过程: 攻击者在一个或多个端点上通过执行 bitsadmin.exe 程序下载和执行恶意代码

#### 假设描述

在我的环境中，有一个敌方用名为 bitsadmin.exe 的可执行文件创建了 BITS Job 作业从而在我的组织中维持了持久的存在

#### 文档记录

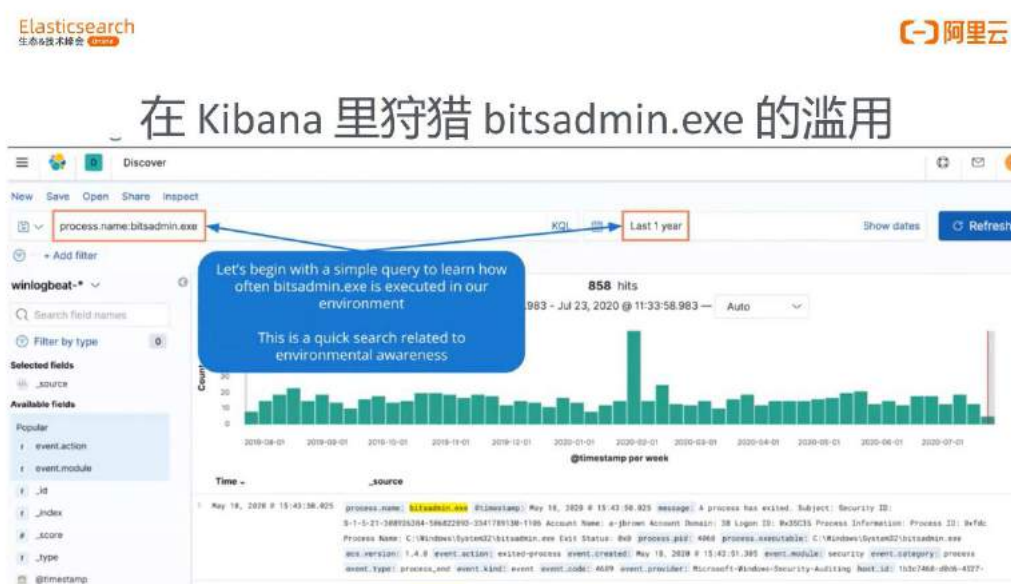
- 时间规划: 2 名团队成员将在 5 个工作日内，工作在这个假设上
- 数据源: 处理监控日志 (Windows event ID 4688) 和 Sysmon event ID 1，团队使用 bitsadmin.exe 在环境中模拟产生 BITS 作业
- 例外的系统和数据源: 无，所有工作范围内的主机都要发送事件日志到我们的 SIEM
- 跟踪技术: 我们使用狩猎团队的 Wiki

首先，我们需要利用 Mitre ATT&CK 给我们提供的 TTP 语言去描述 BITS Jobs。BITS Jobs 本身是 Windows 操作系统后台的一个智能数据，是非常底层的传输服务，它有它底层的一个可执行文件，就是 bitsadmin.exe 这样一个程序，但它的战术是通过滥用这个可执行的下载或执行恶意的代码，通过 HTTP 的方式去解释在我们这个工作环境中 BITS Jobs 的安全威胁。



为了防范和防御安全威胁，我们就要进行威胁狩猎，这也是基于一个威胁狩猎假设的。我们的假设可以描述成在我们的工作环境当中，或者说 IT 环境当中，可能潜在的敌方或者黑客滥用名为 bitsadmin.exe 的程序创建 BITS Jobs 相关的作业。这个 BITS Jobs 相关的作业在环境当中的发生和存在可能会执行相关的恶意程序和代码，这就是我们对这个安全威胁工作的描述。对这样一个安全威胁的狩猎过程，我们会用文档做详细的记录，在这个文档中要非常具体地记录时间或资源的规划，比如我们会计划两名安全管理团队人员在五个工作日之内完成对以上安全威胁假设的狩猎过程，可以利用到的数据源有 Windows 操作系统的事件日志。我们是没有例外的系统或数据源的，希望对我们工作范围内所有的主机都去分析它 Windows 的事件日志。最后，这一次威胁狩猎的工作过程以及结果会记录在我们安全团队所使用的 Wiki 上，这就是我们对整个具体的威胁狩猎工作的描述。

接下来，在执行阶段当中，首先我们要使用 Winlogbeats 去采集每一个 Windows 操作系统中的每一个端点的所有日志。这个采集和归集的时间限度我们希望是最近一年的时间，分析的对象就是这个进程的名字，名为 bitsadmins.exe 的可执行程序的执行次数。我们可以首先在这一年的 Windowslog 日志当中进行一个快速的搜索。



上图就可以清晰地展示出哪些天的执行次数，是怎样的趋势。搜索出这样一个具体的进程结果之后，我们对需要的结果做可视化分析，要用到 Kibana 里面的可视化分析工具。首先，我们可以对这个分析的结果做一个排序和排名，可以排出这些貌似比较异常的作业，我们会去聚焦在 bitsadmins 这个执行参数中包含外部网址的事件当中，也就是说某些机器可



能会被滥用，去执行外部的一些网站上下下载的恶意代码，因此我们就进一步锁定到了在一年的事件中有一次这样的行为。因为这个行为是被具体到了某一次或几次事件当中，所以我们就从大量的实践当中更进一步地缩小了范围，这个事件也给我们带来了更丰富的上下文，我们可以将这次事件具体到哪一个用户所使用的哪一台什么样的设备，那天整个工作当中还有哪些相关的事件。因此这整个事件的结果，一个事件详情的揭露，可能会为本次的威胁狩猎带来一个非常大的收获，这就是我们一个阶段性的成果。

### 三、安全威胁检测规则的管理

基于这样一个非常具体的 Kibana 探测的规则，我们可以编写出 Kibana 里面查询的威胁探测规则的语句。



这个威胁探讨规则语句就如上图中显示的 Process.name=bitsadmin.exe, 并且它这个进程是需要带有五个标志性的参数，这些参数往往就是可疑事件的一些发生。因此，由于我们前期的工作已经探测到这个环境当中确实曾经发生过这种安全入侵的事件，虽然只有一例，但是我们也证明了这个安全威胁在环境当中是存在的。我们在以后就希望通过一个自动化的方式能够通知或提醒到 BITS Jobs 安全威胁事故的复发，因此我们可以基于刚才分析的结果创建一条自定义的探测规则，可以使用 Kibana 当中的 SIEM 这个 app 进入到探测规则的管理界面当中，创建一条叫自定义的查询规则。这个自定义的查询规则一旦创建之后，就可以在后台持续运行。

我们针对 BITS Jobs 的威胁狩猎的阶段性成果就形成了两个重要的成果：

第一、我们刚才所描述的整个过程都会形成文档，记录在安全管理团队的 Wiki 上，这个文档也会公布给所有的干系方，告诉大家我们这个环境当中确实是有这样一个安全事故发生的。

第二、我们会去创建一条自动化的安全威胁风险的检测规则，这个规则在后期会被持续地在 Elastic Stack 这个技术平台上去运行，通过它来预防后续可能发生的 BITS Jobs 滥用的安全风险。这个规则一旦创建之后，还是需要有些维护的，包括安全管理人员可能会多次对这个规则进行测试，确保规则的灵敏性。更重要的是，维护测试也确保了这个规则在今后的安全威胁防护的过程当中是一个长效的存在。

接下来，我们再回顾一下狩猎的基础是什么。



## 数据源，越全面，越好

分层次的高维安全信息采集平台

领域	数据源	特征	工具
网络	PCAP, Bro, NetFlow	实时, 基于数据包	Packetbeat, Logstash (netflow 模块)
应用	日志	实时, 基于事件	Filebeat, Logstash, Sysmon, Winlogbeat
云平台	日志, API	实时, 基于事件	Filebeat, Logstash (AWS Cloudwatch, GCP Pubsub, Azure/Cloud app security)
主机	系统状态, 签名告警	实时, 异步	Auditbeat, Filebeat (Osquery 模块), Winlogbeat, Metricbeat, Heartbeat
活动	扫描, CDN, Web协议	用户驱动, 异步	漏洞扫描器, Heartbeat (TLS 证书检查), CSP 报表, CT日志, CDN日志

狩猎的基础还是安全事件的数据源，我们可以使用 Elastic Stack 当中各种各样的 beats，包括 Packetbeat，Filebeat，Winlogbeat，还有 Auditbeat 这些工具，让我们从网络应用，云平台，主机和网络活动当中收集到这些数据。有了这些数据之后，我们就可以对这些数据进行更进一步的优化。我们希望将任何一条收集上来的裸的安全事件进行处理，这些处理包括了跟威胁情报的关联，IP 地理信息的丰富，以及其他外围系统的查询。

## 通过数据丰富提高威胁情报的质量

基于 Elastic 统一数据定义 ECS



经过丰富之后的这些数据就更具意义，就是一条安全的情报数据。原数据不具备情报的价值，我们把它丰富之后，每一条数据都变成一个安全管理的情报数据了。这些情报数据里面标明了一些非常重要的，基于地理的一些信息系统，来自哪些国家和城市的链接或者网络活动，IP 地址是不是一个已知的非法的 IP 地址，或者已知的恶意的 IP 地址。

有了这些数据之后，我们就可以开展安全分析之旅了。



讲到威胁狩猎这个具体的工作，其实是我们安全分析工作当中一个比较高阶的工作，我们基于安全威胁情报对它进行特定的分析。当然，上文也提到了 BITS Jobs 本来就是 Mitre Att&ck 攻击生命周期框架当中的一条，讲述的具体案例只是通过 Elastic Stack 里

SEIM 的应用将狩猎的工作进行落地, 因此如果想要做这样一个非常高端的威胁狩猎工作的话, 还是需要从最底层开始。通过最底层打通各种数据源, 归集各种数据源, 通过 ECS 的方式统一各种数据源的定义, 并且将裸的安全信息数据进一步的可视化, 把分析的结果提供给安全管理的相关的人员。安全管理的相关人员用 SEIM 这个应用和 SOC 工作流程的其他工具来执行他们安全运维方面的一些日常工作, 比如说安全巡检, 威胁狩猎等。这些工作的结果, 每日工作内容知识的沉淀, 可以汇聚成我们 SEIM 应用当中的一些检测规则。我们比较乐观的一个推测是我们的检测规则可以和实际企业的 IT 环境基础设施更加地匹配, 我们在这个环境设施当中会有意去收集更多的数据源, 从而达到最后的威胁情报集成或用户分析的效果。

所谓的威胁狩猎其实就是在安全分析之旅当中一个比较高阶, 最顶层的威胁情报集成和管理的工作。其实检测规则以及上文所提到的所有内容都还是基于人为对已知攻击模式的检测和判断。除此之外, 我们其实也可以利用 Elastic Stack 当中内置的机器学习功能, 对于我们现有的所有安全数据进行异常检测, 通过它来帮我们探测到那些未知的安全事件的隐患, 这些安全事件的隐患通常会存在于一些异常的用户行为、主机行为、网络行为或应用行为当中。我们通过人工加机器学习的方式, 更加强化了我们安全运维的能力, 可以将威胁狩猎的工作做得更好。

假如要做威胁狩猎这样一个工作的话, 我们还是非常推荐使用这个 MITRE ATT&CK 的这样一个全方位的攻击生命周期的框架, 因为这个攻击生命周期框架当中已经涵盖了两百多种比较流行的, 已知的, 黑客所惯用的一些攻击的策略和手段, 而且这些攻击的模式已经变成一个可用的检测规则, 内置在了我们的 Elastic Stack 这个技术站当中, 也就是说安装了 Elastic Stack 这个技术站后就可以进入我们的 SEIM 应用, 在规则管理当中启用这些两百多条内置的基于 MITRE ATT&CK 的检测规则。你可以基于这些检查规则去检测现在企业当中已有的数据, 也可以用它来做未来威胁狩猎的规划的基础。



基于 Elastic Stack 进行威胁狩猎的工作，我们首先会聚焦在多数据源的关联分析上，将黑客的一些攻击的模式通过已知的安全情报或者模式匹配的方式把它暴露出来，并且通过人工的方式对已有数据元的关联分析，把它所有发生过的一些安全隐患的事故都暴露出来。我们可以基于机器学习，对于一些人无法分析到的，或者我们未知的一些潜在威胁做一个基于机器学习的判断。机器学习不仅是按需去调用的，而且它也可以把我们的一些安全规则内置到它探测检测的过程当中，从而为我们检测出我们可能不知道的一些潜在风险和威胁。

最后总结一下，Elastic 是一家专注于搜索的公司，今天讲到的这个威胁狩猎它的核心能力还是和搜索能力相关。我们希望达到在非常大量的数据源中做近实时的一个安全狩猎的工作。在 Elastic Stack 当中已经内置了 SEIM 应用，这个应用也内置了很多的安全检测的规则，可以非常方便地帮助我们去规划和执行企业当中所需要的这种，针对任意 IT 环境或已有数据的威胁狩猎的工作。Elastic Stack 这个技术栈的话是一个快速发展技术栈，它安全管理方面的探测规则也是在社区里面以开源的方式开放出来的。



# 基于流式计算平台搭建实时分享

分享人：Elastic 中文社区副主席吴斌

视频地址：<https://developer.aliyun.com/live/246152>

本文将通过三个部分展开分析基于流式计算平台搭建的实时分析应用中 Elasticsearch 的实战分享：

- 面向开源产品的架构设计
- Elasticsearch 在流式平台中的角色功能
- 云原生与 k8s 集群管理经验分享

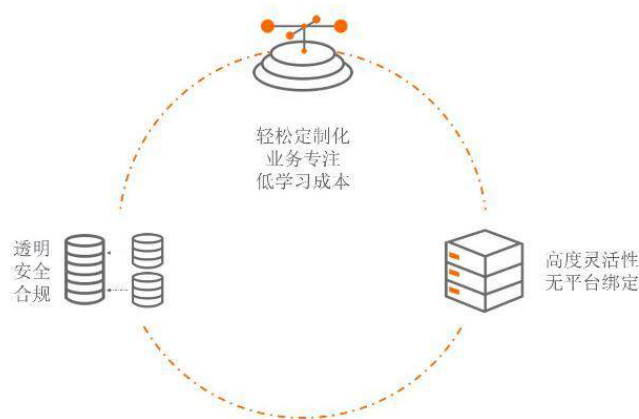
众所周知，Elasticsearch 本身是一个搜索引擎，随着演变和发展，现在它的数据分析能力也非常强大，但它并不是一个万能的引擎，它也需要借助生态的支撑。然后把整个实时分析应用的平台搭建起来也是非常重要的一点，所以今天我们分析的角度是在于 Elasticsearch 本身是如何借助生态的，在这个生态当中自己又处于一个什么样的位置，最后我们会聚焦到 Elasticsearch 本身应该如何去实现快速简单、具备弹性、通用性的一些部署。

## 一、面向开源产品的架构设计

为什么要面向开源产品去做架构设计？很重要的一点是增加平台的通用性和一致性。如果能让整体的架构变得易于维护，不可避免地要借助云平台的一些能力，但又不能跟这个平台完全绑死，这种情况开源产品在这些平台上的一些实现，就是一个更好的选择。



## 为什么面向开源/开放做设计



### 1. 轻松定制化业务，专注低学习成本

面向开源做架构设计，除了它的开放性和灵活性之外，你还能轻松地定制你的产品，我们的开发和运维人员也可以更多地专注在业务本身上，而不是花更多的成本去学习一些商业产品。开源产品的学习成本本身就比较低，因为在互联网上，很多视频网站上，都能获取到很多有用的资源，在中国针对 Elasticsearch 我们也有一个自己的中文社区，这上面不管是针对于初学者，还是对于一些非常深入的大咖们，相信都能有你所需要的一些内容。

### 2. 透明、安全、合规

中国企业逐渐对安全合规的属性变得更加敏感和重视，开源产品能使安全合规这些规则变得更加透明，因为代码都是公开的，能快速地帮助我们通过合规性的一些审核，和安全机制并加固。

### 3. 高度灵活性，无平台绑定

过去 Elasticsearch 本身不能完成所有流式平台的搭建，还需要借助生态。生态上，越多的组件引入进来，系统就会变得越复杂，维护成本就会稳健升高，如果能借助云平台的力量，运维和后期灵活的扩展就会变得相对容易，对未来假设去做移植工作也不会有太大的障碍。

## 4. 【平台组件构成】



我们在搭建流式计算平台时的架构还是非常清晰的，在这里做一个简单的剖析。

首先，数据采集通常来讲是一个分布式的消息队列，它采用的发布或者订阅是一种消息的分发机制，同时还能在后面的计算和存储引擎出现问题时把消息缓存在消息队列里面。这个分布式的消息队列本身也是高可用的，当出现流量峰值，它也能对后面的存储引擎有保护作用。

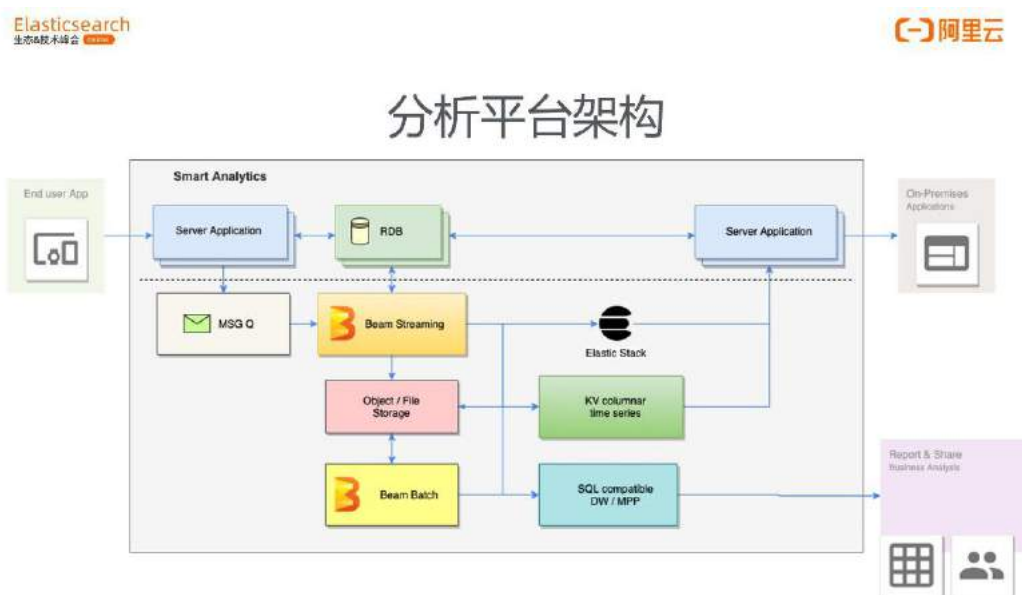
当数据流进来之后，第二步就是分布式计算引擎。首先需要对进来的数据做校验，判断数据是否合法，格式是否是我期待的，有没有脏数据等等，同时还可以去业务数据库引擎里拉取一些数据，把它变得更加丰富，来辅助后期的分析。另外，现在的流式计算平台能让数据计算更加精确，这一点基于事件时间开窗的计算。而流式的分布式计算引擎本身在云上也具备弹性，且可以做热更新，这对整个平台的稳定性和扩展性来说都是非常友好的。

除了消息队列和分布式计算引擎之外，后面我们还会选择三个比较关键的引擎，其中一个就是我们主要要讨论的引擎 Elasticsearch。除此之外可能还会引入高 IO 的 KW 列存，最常用的是 HBase。这两个引擎在这个组合里面扮演的是对于非常热的数据的一些处理和计算，是实时的一部分。第三个引擎我们引入的通常来讲是线下的比如数据仓库或者一些 MPP 引擎。大家的选择面是非常广的，比如传统的 HIVE，还有开源的 Greenplum 等等，甚至一些商业产品都可以是在我们选择清单里的一些存储，这些存储的角色更倾向于一些偏线下的计算，或者是比较温的一些数据的计算，比如过去七天的一些数据的计算，基于每天

去绘制的报表和计算。在热数据这一部分，Hbase 通常来讲不是一个非常必要的引擎，只是在我们有非常高 IO 的吞吐的场景下不得不引入它，它的核心的实时数据分析这一部分通常来讲我们还是用 Elasticsearch 去完成的。

基于上面这一套引擎，下面还会有两个非常重要的基础组件，这个在云端实际上也都给大家封装非常好的一些成熟的产品和底座。如果大家是在 on-premise 自己的 IDC 机房里去部署的，还需要维护这两套系统，在本地需要有一个高性能的网络，能够支撑比如消息队列之间的消息传递，还有分布式计算引擎之间 worker 节点的一些 shuffling 的操作，也都需要通过高性能网络及交换数据，数据引擎之间的数据交换，节点之间的 shuffle 也都需要高性能的网络来支撑。

另外就是底层的分布式存储，这个存储的选择当然也非常多，在云端我们通常会借助对象存储来做比如分布式计算引擎的文件输出，或者数据的输出，还有错误数据的一些落地，后面的数据引擎比如 Elasticsearch 就会做一些数据的备份，比如说 snapshots。分布式存储可以作为我们数据的一些 archive，比如一些老旧数据的备份，或者在关键时刻需要重塑，可以很快地从这些 snapshots 里面恢复数据。



这个平台的组建实际上是非常简单的，那么在根据这些组件搭建真正的架构的时候，就可以看到上图这样的一张架构图。中间虚线部分上面是真正的业务引擎，里面有我们的服务器，还有处理业务的关系型数据库。真正的数据就是从我们服务上来的，通常来讲，比如说可能有这些服务器的日志，监控，业务上来讲，从这个业务服务器上我们可能会采集很多用

户端的一些行为，比如说用户在你的平台上购买什么样的商品，或者社交类的产品、内容类的产品都点开了哪些内容等等。这些数据就会被实时地发送到刚才介绍的消息队列里面（图中左边信封的图标），在这个队列缓存里进行缓存之后，就会把它放到后面的 Streaming 引擎里面去。这个引擎用到了 Beam 这样一个驱动层，分布式计算的一个框架，它也是一个开源的分布式计算引擎的框架，只是一个驱动层。在这一方面，我们想把这个平台打造的通用性好一些，所以在这个关键的分布式计算环节，我们选择了这样一个驱动层。它所带来的好处是比如我在一次编码之后，可以驱动不同的分布式计算引擎去帮我完成计算任务，而且它是批流一体的引擎，Beam 的代码就可以驱动 Flink 去完成计算任务，也可以驱动很多其他的 driver。你可以去做流式的处理，也可以做线下的 Bash 处理，这是我们选择 Beam 的一个原因。当然，Beam 也会有它一定的局限性，可能每个引擎之间有不同的非常独立的 calculator，没有能很好地移植到 Beam 里面，它就不支持。所以大家在真正使用的时候，还是要根据自己的业务看 Beam 是不是一个非常好的选择，目前在我们服务的客户当中 Beam 基本上是可以满足绝大多数的需求的。

既然 Beam 是一个批流一体的引擎，并且分布式计算已经会做一些 ETL，然后从我们的业务数据里我们可能会拉取一些业务表，来跟我们实时采集的用户行为数据或者日志数据做一些联动，方便我们日后去做分析。这个时候我们会把一些不符合或者有问题的数据输送到对象存储上面去做备份，与此同时还会把明细数据也都打包压缩备份一份放在对象存储上面以备不时之需。

接下来后面接的引擎我们就会把一些实时的数据直接注入到 Elasticsearch 引擎里面，去支撑我们的业务包括实时分析业务。在非常高吞吐的情况下，我们会引入 HBase，就是下面介入的 KV 型的列存。它的列存表的设计有宽表或高表两种，是完全取决于我们的业务的，但是一旦我们引入 Hbase 这种 KV 引擎的话有一点需要注意，就是明细的数据才可能会录到 HBase 里面去，为了减少 Elasticsearch 本身的 IO，我们会把一部分计算任务放在中间的分布式计算引擎里面去做。举个简单的例子，假设打车这样一个场景，用户或车辆的实时地理位置信息是非常海量的数据注入到系统里面，在这个订单当前没有完成的时候，我们都可以把这些实时高 IO 的数据放到 HBase 里面进行查询，或支撑你的业务，当这个 session 结束之后，把它归拢成一条数据放到 ES 里面去，我们在 Elasticsearch 里面就可以做一些实时订单的分析，轨迹，金额，或者官方的一些客服人员去对这个订单做查询和服务我们的客户。这就是 ES 加 HBase 在实时的场景是这样配合工作的，并不是同样的数据全部录入到两个引擎里面去。

第三个与此同时我们还会做的事情就是放到 Data Warehouse 里面去做一些线下的分析。另外，可以看到 Beam 还会承担一部分 BASH 的工作，也许我们会有一部分这种批处理的工作，在实时的数据落到对象存储之后，我们也可以定时地让 Beam 去驱动 Flink 等引擎完成后面的一系列批处理的工作。

## 二、Elasticsearch 在流式平台中的角色功能

这个平台架构实际上也是非常清晰的，在这个平台里面我们再具体地去看一下 Elasticsearch 到底承载一个什么样的角色和哪些功能。



### 1. 文本检索

首先它最主要的功能就是文本的检索，他适用的场景主要是日志，运维，开发，还有一些实时业务的客服，客服可以根据这个日志快速找到一些订单的线索。

### 2. 已知数据的计算

第二个功能是已知数据的计算，是一些实时指标的计算，固定的报表，实时的大屏展示等等。这些数据通常来讲，schema 是我们已经设定好的，我们知道后面根据哪些指标维度进行分析，比如把它做成一张报表或者做成一个大屏等。

### 3. 未知线索探索

另外一个非常重要的 Elastic 的特点，就是对未知线索的一些探索，这也是为什么我们要在实时业务中使用 Elastic search 的非常重要的原因，因为 Elasticsearch 非常的快。

当你发现业务当中有些异常的指标出现的时候，你会拿到这个线索，但你并不知道是由于什么原因导致的，可能是完全未知的一种攻击手段，并不能通过规则引擎去发现它。这个时候我们需要拿着这个线索作为一个输入，在实时的数据里面去探索，甚至结合一部分我们的历史数据，快速地进行一些复杂的过滤和筛选，然后找到跟这些数据相关的其他可能的维度和数据，甚至很快能够找到一些归因，这个是搜索引擎在做数据探索过程当中非常重要的一个特点。

所以当你的业务或分析平台需要有这样的需求时，就可以从这三个维度去思考，是不是有些实时要检索的文本，是不是有一些已知的报表的计算要展示，或者说会不会要对一些未知的线索进行快速的探索，这些都是 Elasticsearch 本身的一些功能和特点。

## 三、云原生与 k8s 集群管理经验分享

### 1. Elastic Stack 部署方式

首先我们看一下 Elastic Stack 的部署方式，基本上是以下三种。



过去我们常用的是最左和最右两种，左边的这种是我们自己去部署，不管是在云平台的虚拟机上，还是在我们自己的机房里面，它的优点显然是非常通用，一致性非常好，也非常透



明，你甚至可以编译自己的 Elasticsearch 去部署。它的缺点也是显而易见，比如大集群或集群多的时候，它维护和升级会相当困难，包括多集群的升级和数据之间的导入导出等等，并且当它出现错误时，恢复周期会变得非常长且复杂。这个对于大集群来讲不是一个推荐的方案，小集群十个节点以内就还可以。

再看最右的方案，就是以 SaaS 的部署方案，它的优点也是非常显而易见的，在这个浏览器里轻松地点一下就可以完成集群的一个部署和后期的运维，甚至包括升级和数据之前的导出等等。它的缺点是它的细节不透明，并且网络的一些拓扑结构，甚至自己的网络都会受一些限制，集群的入口网关的性能也是受限的。所以 SaaS 的好处就是非常简单，但是后面的定制化，甚至灵活性，甚至第一时间能不能得到 ES 的升级来讲可能都不是最好的一个选择。

根据我们去年一年的经验来看，Kubernetes 上的部署会是一个非常好的选择，并且也比较成熟，这个取决于 Kubernetes 上支持的两个好的技术。

最早期实际上 Kubernetes 并不是对这种数据层产品非常友好的平台，因为它都是无状态的应用。随着 operator 概念的引入，它才把整个数据平台放到 Kubernetes 上，变成一个具有可执行性的方案。再加上 ES 本身自己的设计，反而让 Elasticsearch 本身对 Kubernetes 平台是非常友好的。它的优点也是显而易见的，后面会用一些典型的场景去展示。另外一点是 Infrastructure as code 的概念，你可以把你集群的描述放到一个压缩文件里，它就变成一个你的集群部署的结构了，后期只需要改变这个样本文件就可以改变你的部署和架构等等。第三点，在 Kubernetes 上的弹性非常好，只要资源够，你可以瞬间发布出大量 Elasticsearch 的节点。而且相对于 SaaS 的方案来讲它的资源也是独显的，不会出现你不知道你的容器是否跟其他的容器部署在一个物理主机上，他们在共享、竞争着 CPU 的使用等等场景。

Kubernetes 固然好，但是使用率并不高，主要是因为在一般的中小企业里面引入 Kubernetes 从某种意义上实际上是给自己平添了不少麻烦，因为不管是应用层还是数据层，都需要更多地去维护一套 Kubernetes 系统。因此，这个可能简化了 ES 和上面应用的使用场景，但是对你的运维团队带来了另外一个维度的负担，那就是对于 Kubernetes 集群的维护，所以我们也是比较推荐大家在云上面去做 Kubernetes，使用云端托管的 Kubernetes 服务来部署 Elasticsearch。目前来讲，我们社区维护的 Kubernetes 这个版本首先还是受限于 Kubernetes，因为没有 Kubernetes 就部署不了。第二，目前还只

是一个开源的版本，ES 官方在未来应该会逐步推出商业版本，大家可以时刻持续关注，我们也会第一时间来更新社区。

## 2. ES 集群架构的预置

  
生态&技术峰会 2024



预置集群架构

zone-a	zone-b
ES x 2	ES x 2

zone-a	zone-b
Master x 2	Master x 1
Ingest x 1	Ingest x 1
Data x 2	Data x 2
Coordinating x 1	Coordinating x 1
ML x 1	on any available node

灵活存储方案

- dingo-pdssd 高性能  
type: zonal SSD  
best for: Data nodes (hot/warm), Master nodes, ML nodes
- dingo-pdssd-ha 高性能高可用  
type: regional SSD  
best for: Master nodes, Data nodes (warm/cold)
- dingo-pdssd-balancer  
type: zonal balanced SSD  
best for: Data nodes (warm)
- dingo-pdssd-blanced-ha 中等性能高可用  
type: regional balanced SSD  
best for: Master nodes, Data nodes (warm/cold)
- dingo-pdhdd 磁盘  
type: zonal HDD  
best for: ML nodes, Ingest nodes, Coordinating nodes, Kibana, APM, Data nodes (cold)
- dingo-pdhdd-ha 高可用磁盘  
type: regional HDD  
best for: Data nodes (cold)

这里我们给出一个场景，在后面给大家预置的项目里面可以看到我们对 ES 的集群架构做了一个预置。我们提供了三种方案，第一个是单节点的，通常只是对开发人员的，可以让整个开发，测试，还有线上的环境完全保持在同样一个版本。第二个是全角色的节点（左上角第一张图），这里面我们对于每个节点做了一些深度的优化，不同角色的节点都应该有哪些相应的参数的调整。可以看到，在全角色的部署上面我们把它分布到两个区里面了，如果你是在自己机房里面的 Kubernetes 上，你可以把它当做两个机架，我们对于数据的分布做了一个高可用，也就是说我们会让你的主分片和 replica 分别分布在两机架之上。

除此之外，我们还封装了一个分角色节点的部署（右下图），这个通常可能会应对一些比较大规模的集群，就是把主节点，数据节点，还有 coordinating，Ingest 界面全部分离出来，这样做的好处就是当集群遇到高峰数据的时候，你可以相应地去调整 Ingest node，Coordinating node 或是 Data node，所以这个集群的弹性是存在的。另外我们还给大家预置了很多存储的方案，不同平台提供的底层块存储或者磁盘是不一样的，所以说这里需要根据自己所选用的平台做一些灵活的调整。

## 3. 恢复策略

Elasticsearch  
生态&技术峰会

阿里云

## Failover 策略与恢复



左图是数据在计算平台上的一个有向无环图，数据从上面的节点流进来之后，经过后面的计算节点做一些简单的 ETL，最重要的是下面的两个输出，左边的输出是错误数据和明细数据的备份的输出，右边的数据是计算完整理后的一些数据或者是聚合后的一些数据，把它输出到 ES。左边这张图就是分布式计算引擎在做的事情。

Elasticsearch 本身就是中间这张图，我们会跟存储做一个关联，不管下面是什么样的存储方案，还是云上的对象存储，我们都会定期去做 Snapshot。

有些人可能会提出疑问：左边备份了明细数据，中间为什么还要去做 Snapshot 呢？是否有些多此一举？

实际上是这样的，我们的明细数据备份之后可能不光是给 ES 去用，由于 ES 本身的 Snapshot 也是定时去做的，有些人可能一天做一个增量备份，有些人可能一小时做一个增量备份，假设今天这个时间节点出现问题，我用 Snapshot 恢复只能恢复到昨天的数据，所以说我在真正重建这个集群快速恢复我的数据的时候，在有必要的前提条件下，是 Snapshot 加明细这样一个恢复的策略，能让我的数据恢复到刚才集群出问题的状态。这些消息都会缓存或者阻塞在分布式消息队列那一层，所以大家完全不用担心。

右上角代码的截图是我们在 Kubernetes 引擎给大家做了一个叫 Snapshotter 的容器，它会定时给 Elasticsearch 做一个类似快照的 URL，endpoint，然后定时地把数据备份到你的目标存储上面去。但是最新版本的 Kibana 里面我们现在好像也可以通过

policy 去做这件事了,也就是说现在 Kibana 或者是 ES 本身自己的功能确实越来越丰富了。右下角的截图大家可以看到,我们封装好了 Elasticsearch 的 docker,自身部署到 Kubernetes 上的 container 可以在初始化容器的时候去装一些相应的插件,比如用来对接交换数据的对象存储或 HDRS 插件等等。所以如果大家有需要备份到云平台的某个对象储存,你需要去找这样的插件然后把它安装到 ES 里去,就可以通过 Snapshotter 做一个定时备份。

## 4. 集群架构调整

除了恢复之外,在 SaaS 里面另外一个我们经常做的操作就是对集群的架构进行调整。



调整节点规模

```
kind: Elasticsearch
metadata:
  name: elasticsearch
spec:
  version: 7.10.2
  #http:
  #service:
  #spec:
  #type: LoadBalancer
  secureSettings:
    - secretName: gcs-credentials
  nodeSets:
    - name: zone-a
      count: 20
      config:
        node.master: true
        node.data: true
        node.ingest: true
        node.ml: true
        xpack.ml.enabled: true
        node.store.allow_mmap: true
        index.store.type: hybridfs
        cluster.routing.allocation.awareness.attributes: zone
        node.attr.zone: asia-east1-a
        cluster.remote.connect: true
        xpack.security.authc.anonymous.roles: monitoring_user
  volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data
```

集群架构调整

```
nodeSets:
  - name: zone-a-ingest
    count: 1
    config:
      node.master: false
      node.data: false
      node.ingest: true
      node.ml: false
      xpack.ml.enabled: true
      node.store.allow_mmap: false
      index.store.type: fs
      cluster.routing.allocation.awareness.attributes: zone
      node.attr.zone: asia-east1-a
      cluster.remote.connect: true

  - name: zone-a-master
    count: 2
    config:
      node.master: true
      node.data: false
      node.ingest: false
      node.ml: false
      xpack.ml.enabled: true
      node.store.allow_mmap: false
      index.store.type: hybridfs
      cluster.routing.allocation.awareness.attributes: zone
      node.attr.zone: asia-east1-a
      cluster.remote.connect: false
```

首先左图是我们在调整节点或规模的时候,比如我想在当前的这个集群里面把我的节点从两个调整到二十个,这个动作如果在 Kubernetes 集群资源足够的前提条件下,基本上一分钟之内就可以完成,其实跟节点的多少关系并不是很大,当然越多可能会稍微慢一些,总体还是非常快的。你只需要改一个数字,直接 apply 你的部署,集群的规模立刻就可以增长上来。但是需要注意的是,要缩小这个集群可能会慢一些,不过它至少不是一个手动工作,且是由 ES 官方的 operator 帮你逐个把节点拿掉的,这个动作一定要非常小心。首先是要做数据的迁移,把当前的数据移到其他的节点上去,做一系列的 check 之后再把这个节点拿掉。但对于我们这些用户来讲,我们只需要简单地改一下数字就可以。

同理，升级也是，左图大家能看到 version 7.10.2 现在已经是最新的版本了，如果未来有新的版本 release 的时候我们也只需要改一个版本号，你的集群会通过 operator 自动升级。所以调整架构规模是非常简单的一件事情，如果在调整过程当中出现了任何问题，你也可以选择去重建整个集群，当然数据越多需要的时间也会相对更多。

右图是架构的调整，上面是数据 Ingest 的摄入节点，下面是主节点，与左边的节点调整同理。比如从一个全角色的节点引入 Ingest node 然后把 master 节点分离出来，实际上你需要做的只是增加更多的 node sets。在 node sets 下面我们定义 node set 在 Kubernetes 里面扮演什么样的角色，分别有 master 节点、热数据节点、冷数据节点等，都可以通过 Elasticsearch 去做一些标记，然后在 node set 里面把它分离出来。我们在实际线下的操作过程当中，会根据实际情况进行调整。比如有些用户刚开始给 20 个集群做了角色分离，后来他发现了集群的性能受限，有太多的节点去扮演 master，ingest 或者 coordinating，他们的压力并不高，数据节点偏少，这时我们就会让这 20 个节点全部变成数据节点，变成同样一个角色，来提高整个集群的性能和利用率。

还有一种情况，假设随着同等节点的个数调整，比如从 20 个调整到 40 个，你发现 utilization 的不均匀之后又想把它分离出来，同样没有问题，也是通过 node set 的模式去调整就好了。

## 5. 专用网关

我们为 Elasticsearch 量身打造了一个专用的网关，因为不管在线下操作还是在 SaaS 做都会有非常大的限制，但是如果你在 Kubernetes 里面做，Kubernetes 自身有非常好的预定义的 ingress，ingress 本身自己也会有一些限制。

我们给大家打造的 Elasticsearch 专用网关具备以下特性：

- 高可用，不停机索引，自动处理后端 Elasticsearch 的故障，不影响数据的正常摄取。
- 写入加速，可自动合并独立的索引请求为批量请求，降低后端压力，提高索引效率。
- 查询加速，可配置查询缓存，Kinbana 分析仪表板的无缝智能加速，全面提升搜索体验。
- 透明重试，自动处理后端 Elasticsearch 节点故障和对查询请求进行迁移重试。
- 流量克隆，支持复制流量到多个不同的后端 Elasticsearch 集群，支持流量灰度迁移。



- 一键重建，优化过的高速重建和增量数据的自动处理，支持新旧索引的透明无缝切换。
- 安全传输，自动支持 TLS/HTTPS，可动态生成前证书，也可指定自签可信证书。
- 精准路由，多种算法的负载均衡模式，索引和查询可分别配置负载路由策略，动态灵活。
- 限速限流，支持索引中限速和限流规则，可以实现索引级别的限速，保障后端集群的稳定性。
- 并发控制，支持集群和节点级别的 TCP 并发连接数控制，保障后端集群和节点稳定性。无单点故障，内置基于虚拟 IP 的高可用解决方案，双机热备，故障自动迁移，避免单点故障。
- 请求透视，内置日志和指标监控，可以对 Elasticsearch 请求做全面的数据分析。

## 6. 相关资源

关于如何落地，这里给大家提供了三个项目。

- Elasticsearch on k8s (中文社区维护)
  - <https://github.com/elasticsearch-cn/elastic-on-gke>
- 流式分析平台框架 (Beam)
  - <https://github.com/cloudymoma/raycom>
- 极限网关 (持续完善中)
  - <http://gateway.infini.sh/>

中文社区里维护的项目 Elasticsearch on Kubernetes，如果大家想快速地在 Kubernetes 上部署 Elasticsearch，不管你的平台是什么样的，都可以参考这个项目，或者里面有很多预置的脚本，你可以来实现快速的部署。

流式计算平台搭建所在 Beam 的执行方案，我们在这里写了一个 Beam 的框架代码，有了这个框架之后，很快就可以把一个非常复杂的流式计算任务或者一个平台搭建起来，真正在做开发的时候只需要填充中间的业务逻辑和一些相关的代码就可以了，这是我们构建这个项目的目的。

最后一个是在持续完善中的极限网关，欢迎大家去试用，如果大家想知道更多的信息，欢迎到我们的中文社区里面来留言。



# Elasticsearch 基于 Pipeline 窗口函数

分享人：力萌信息数据技术专家李猛

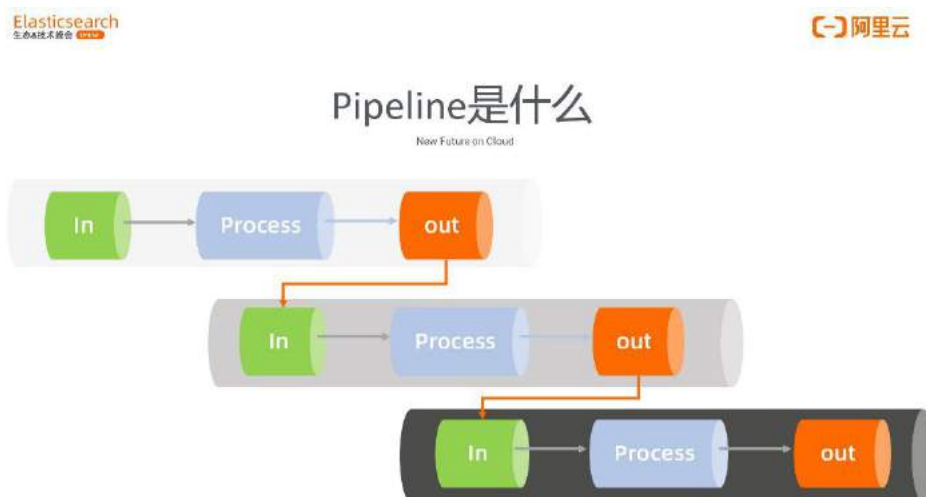
视频地址：<https://developer.aliyun.com/live/246153>

本文将通过三个部分展开介绍 Elasticsearch 基于 Pipeline 窗口函数如何实现实时聚合计算：

- Pipeline 实时计算模型
- ES-Pipeline 实时计算能力和工作特点
- ES+X 实时计算畅想

## 一、Pipeline 实时计算模型

首先，我们来探讨一下 Pipeline 实时计算模型是什么。



Pipeline 翻译过来是管道的意思，上图大家可以看到有三根管子，我们在大数据领域或者在做应用系统的时候，其实编程抽象来说就是这三点：输入数据，process 处理，输出。如果业务程序或者大数据逻辑比较复杂，那么输出就会成为下一个管道的输入，所以就会到第二根管子；第二个管子处理完之后又会到下一根管子，也就是不停地在写很多复杂的数据处理逻辑。其实早期很多同学去做大数据开发，Map-reduce 如果把中间的数据描述

出来会发现也是这三步。Map-reduce-map-reduce 的循环，中间的数据一直在，也是按照这种管道的模型不断地在变化。其实最早 Pipeline 的思维是来自于早期接触到 Elastic 公司推出的 ELK 三件套，其中采集数据的编程的模型配置里面有三个步骤其实就应对了这个模型，在编程里面可以 input，然后可以接受上一次处理的逻辑，这就是管道。

## 1. Streaming 计算模型

我们在大数据领域经常会谈到一个概念叫流式计算，而 Pipeline 这个模型无论是做实时计算还是离线计算，其实思维都是 Streaming 的计算模型。

Elasticsearch  
生态&技术峰会

阿里云

### Streaming 计算模型

New Future on Cloud



如图，DStream 数据从上游进行到下游，是按照追加这种方式的一种流计算，我们要去从中提取数据进行一些模型和处理。而在 SPARK 领域，它目前的流计算只支持微批处理这个概念，但这并不影响我们管道实施的思维模型。

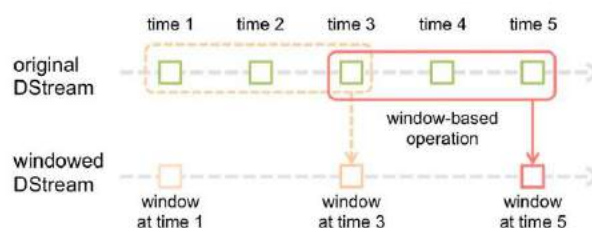
## 2. Streaming 窗口计算模型

Elasticsearch  
生态&技术峰会

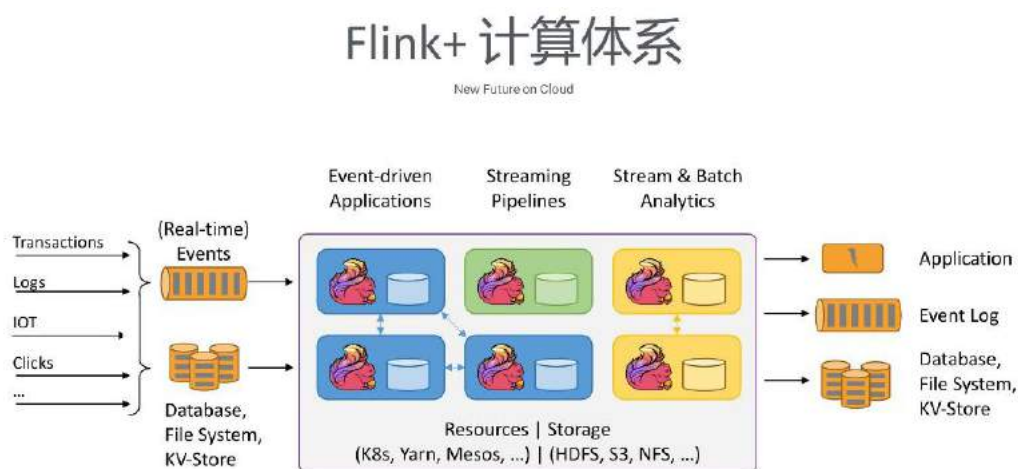
阿里云

### Streaming窗口计算模型

New Future on Cloud



在 Spark 中我们又可以做一个窗口计算模型，创建一个 window，设定一个 1-2 分钟的时间窗口，在时序的数据处理场景确实很合适，比如使用 spark streaming 等等。当然有很多数据场景上它并不是时序的，这个时候我们要基于别的逻辑去做这种数据的窗口计算，我最早的窗口思维也是来自于 Spark，大概四五年前系统地学习 Spark 的时候，第一次感受到 window 其实就是我们传统地做数据统计的时候设定的一个时间范围，只是换了一个计算引擎。以前我们是在数据库里选定一个时间窗口，在 Spark 里我们是基于一个内存模式把数据丢进去，然后在这个范围内计算。



当前在实时计算领域，Flink 绝对是领先的。大家可以看到，在 Flink 计算体系里，中间是 Flink 的计算引擎，它的上游可以支持很多日志，IOT，Clicks，real-time，离线型的 Base 事件，中间它会通过一些管道处理，这些管道本身就可以支持互相的穿插，通过管道把一些数据处理完后输出到另外一个管道，做下一步的数据。如果数据最终没有再继续处理，就把这个数据交换出去，存到某一个系统里面。

图中大家可以看到，用 Flink 做实时计算至少需要三点：

第一是上游的数据输入源 input，中间就需要用 flink，当然也可以选择自己写入程序，如果简单的话是没有任何问题的，或者用 logstash 等等。下游的输出端也有很多，可以输到下一个应用程序里面，可能是另外一组的 Flink 或其他的系统，也可以输入到 Event Log，也可以输出到最终的数据库里面。但我们想讲的 Pipeline 并不是这个意思，这只是我的一个思维起源，Flink 这个框架产品帮我解决了很多调度的问题。

Q：现有流计算的问题是什么？

在标准的实时计算的领域，至少需要三个以上的步骤。

Elasticsearch  
生态技术峰会

阿里云

## 现有流计算问题是什么？

New Future on Cloud



第一，现在基本上大部分企业都会选择 Kafka，中间选择 Flink，下游根据应用程序逻辑有可能还需要下一步的处理，会放到 Kafka，也可能最终不需要处理，需要给 Elasticsearch 做最终查询。目前来说 Elasticsearch 在数据查询方面是领先的，在这个领域里是最好用的。我们在做一个标准的实时流计算的时候，基于 Pipeline 会需要输入-处理-输出，也就是去搭建一个哪怕是轻量级的实时数据处理计算都需要融入至少三个产品，会带来什么问题？大家可以想象，在 IT 里做系统架构等等，每增加一个处理的环节，增加一个节点，或者每增加一个大型的数据系统融入进来，这个系统的复杂性就会增加好几倍，并且数据架构的可靠性也会降低。同时，这三个产品对于研发人员和对架构师的考验非常大，不能保证很快就可以完成。

举个例子，我们一般数据计算之后，会用 Elasticsearch 去存储，而 Elasticsearch 也有多方面的配置，索引的创建，集群的搭建等等又回到一个现实的问题，就是数据库是否需要一个 DBA，上面有什么业务在并行，所以每一次当我们融入新的技术都会遇到问题。所以今天探讨的是，虽然 Pipeline 计算模型非常好，但是现在的实时计算的思维是有问题的，一些产品并不能简化我们的操作。

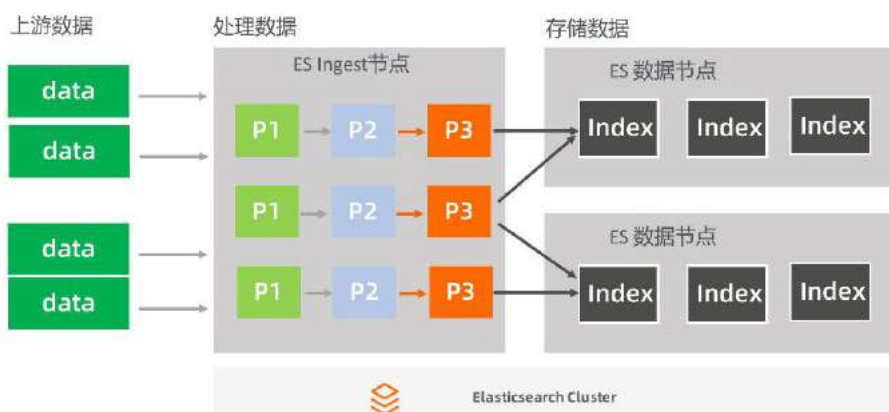
## 二、ES Pipeline 实时计算能力

ES 基于它已有的特性，为了避免上面三件套的问题，在 Pipeline 上做了一些工作。

### 1. ES Ingest Pipeline

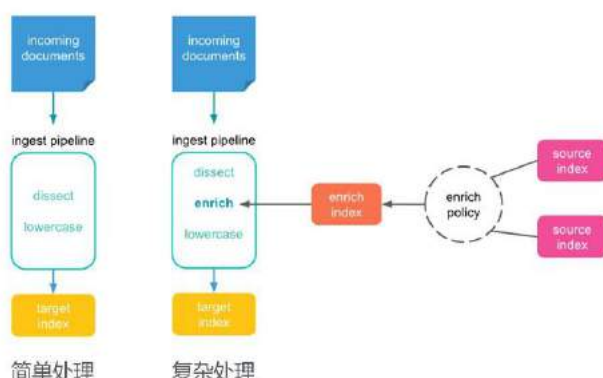
## ES Ingest Pipeline

New Future on Cloud



Elasticsearch 推出了 Ingest 功能，即数据处理。图中大家可以看到，上游数据用任何输入源都是可以的，然后经过 Elasticsearch 里 Ingest 这个具备处理能力的节点，让你的数据经过 Pipeline 管道的处理，可能需要拆分数据做一些复杂的计算。比如本来传入一个性别，需要把性别转换成矩阵，把男和女转化为零和一这样的区别，然后就可以在这里面去编程，完成之后数据直接输出到一个索引里。图中大家可以看到，相比之前的基于标准三件套的实时处理要简化了很多，用一个产品可以替代之前的几个产品。可以把 Kafka，Flink 的逻辑合并掉，全部用 Elasticsearch 来替代，就更加便利。所以如果你对 ES 有兴趣或是熟知的话，可以大规模地使用它，它也有很多缺点，但是它非常的实用，用一个技术替代了三个技术，从图中可以看到它其实也是完全符合 Pipeline 聚合模型的。

## ES Ingest Pipeline

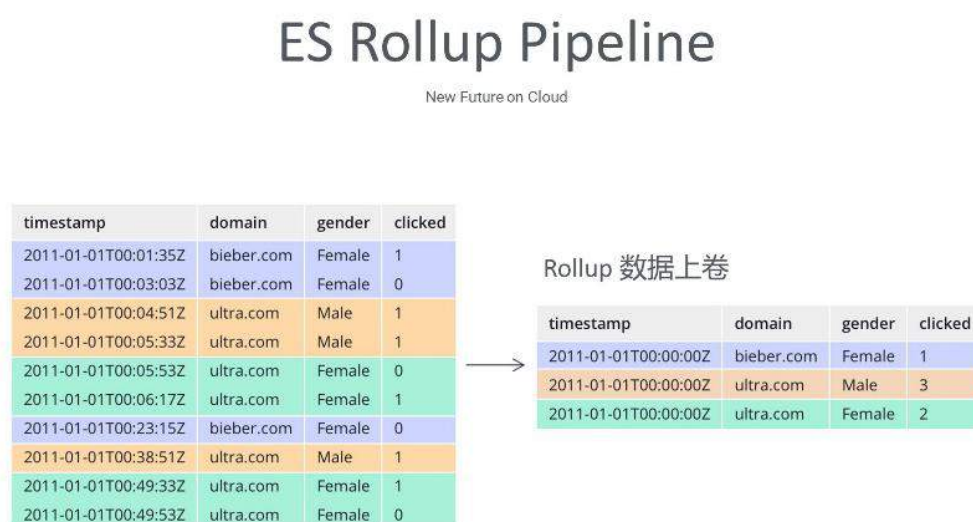


```
PUT _ingest/pipeline/my-pipeline-id
{
  "description": "describe pipeline",
  "processors": [
    {
      "set": {
        "field": "foo",
        "value": "bar"
      }
    }
  ]
}
```

创建管道

Ingest 里提供了很多 pipeline 的函数，其中有一个功能叫 enrich。当数据输入后，ES 的索引可以做一些反查比对，然后再把数据丰富一下。相当于数据进来后要去查一下数据库，然后把一些附属的属性绑定，最终塞到另外一个索引里，其实这也是一个标准的管道。所以在现实世界里管道编程的这种思维其实早已深入人心，并广泛应用。右图是来自于一个标准的函数，这个函数里还有一个用来给原数据增加一些字段的 processor，这个是 ES 的一个特性。对于这些特性不用去刻意地记录，当你认可我们这种编制思维模型的时候就可以去学习一下。

## 2. ES Rollup Pipeline

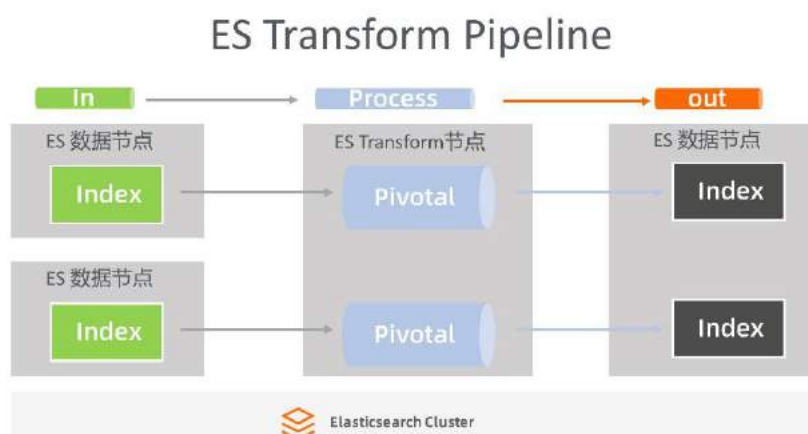


Rollup，即数据上卷，图中左边大家看到数据有三种颜色，这些数据表示的是一个点击事件，记录某一些用户点击的事件，性别。在原始的数据里，我们可以想象，假设做一个网站或系统，pv、uv 的统计，每天可能有上 10 亿的数据量，但是我们分析需求的时候实际上只需要分析今天什么时候哪个域名点击了多少次，这个我们可以通过 Rollup 这个概念把数据做一次转换，压缩。到右边，我们的数据就已经精确到天或者按小时。对于 ES 来说，它其实也是遵循三个关键步骤：第一，原始数据输入到 Elasticsearch 里存起来，中间开启一个 Rollup 实时计算的能力，然后把数据经过一定的折叠之后，输出到另外一个索引，这样就完全满足了 Pipeline 的思维，也完全满足了实时计算。所以在这个领域，ES 算做了一个伟大的创新，只需要一套去处理，就可以把管道模型深切地融入到自己的数据里。如果基于 Flink 做数据统计，上游会先用 Kafka 输入数据，然后中间用 Flink 计算，比如每 1000 条，每 1 万条数据就 Rollup 一下，然后输出到另外一个下游又会放到 ES 里来存储，这就会带来技术的成本，实施的代价。



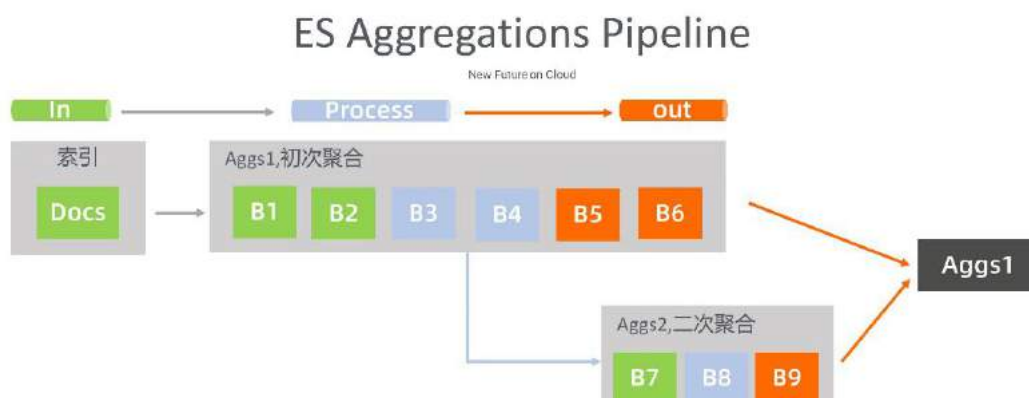
我们用颜色解决了海量明细数据的查询，但是现在又有一些隔夜的统计需求，比如定期的，该如何去做？常规来说，有些大数据的专家给你建议可能会需要把数据取出来，这个其实并不是很好，所以这个时候如果你用的是 Elasticsearch 这一套件，可以自己创建一个 Rollup 节点输到另外一个就完成了。带来的经济效益不言而喻，而且功能特性也非常简单。

### 3. ES Transform Pipeline



Rollup 其实是基于时间的维度，那么如果数据并没有时间，想要做 Pipeline 转换的处理也是一样的。在 ES 里推出了功能叫 Transform Pipeline 的模型，同样可以解决这种问题，只是统计方式略有不同。在 Transform 里用户也可以编辑很多自己的函数，也可以写一些复杂的脚本。

### 4. ES Aggregations Pipeline



接下来讲到 ES 强大的聚合能力，聚合其实是做统计，比如 1000 条数据要做一个 SUM 的值，或者 AVG，Min，Max 等等，这些就叫 Aggregation。在 Elasticsearch 里其实是会根据现实世界融入自己的一些逻辑。图中我们可以看到，聚合一共分成了两次，初次聚合基于原始索引输入做一次聚合，聚合之后压缩，比如原来有一亿条数据，聚合出来大概的结果就只有 1 万，然后再基于这 1 万左右的数据又要做二次聚合。在 ES 里这个思维就叫 Pipeline，可以在官方网站搜索到叫 Elastic Pipeline 的聚合。二次聚合之后，ES 把第一次和第二次聚合的结果一起推给应用端，这个特性是其他很多数据产品没有的。比如原本数据库里的 SQL 是做不到的，其他的一些大型产品可能会用到，但是复杂程度极高，所以比较推荐 ES 这个功能。

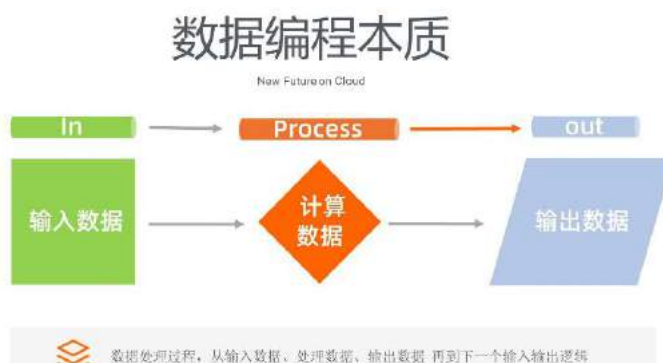


在 Elastic 里，Pipeline 的窗口函数其实提供了很多方式，比如 Aggregation 目前提供了至少图中这三种方式。比如可以选择 moving avg 计算移动的平均值，通过自定义函数 moving fn 写自定义脚本等等。

以上就是基于 Pipeline 实现窗口计算的一些现实的问题和逻辑，还有 ES 的一些特性。

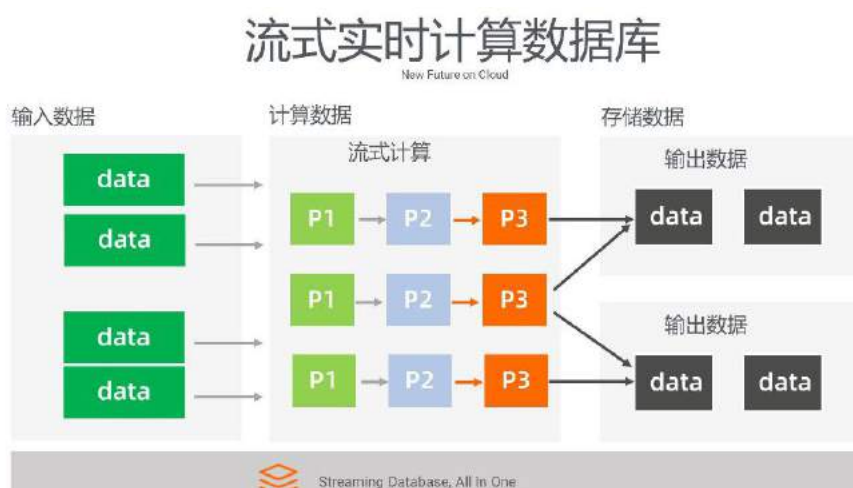
### 三、ES+X 实时计算畅谈

#### 1. 数据编程本质



现实世界中，写一个函数，做一个系统，会发现这个系统提供对外一些 API 输入数据，然后系统做一些逻辑，逻辑之后又输出，可能输出到下一个处理环节，如果是数据库就存在数据库里面。输入的地方也是一样，假设是一个 rest API，接受外围的请求，处理，输出，返回数据，假如是一个大数据里面的离线计算批处理，也是写一个批处理逻辑输入数据，比如 Spark 读取文件，然后做计算，输出，可能又存到输入上去了，或者是输到 ES。如果这一步完成了，我们在大数据里面经常还要做调度编排，一般会用 Airflow 去完成，在编写 Airflow 流程的时候，实际上是编写一个一个流的方式，然后把它挪到下面去，再输入到下一个逻辑。

## 2. 流式实时计算数据库



虽然 ES 也具备一些实时的处理能力，但它有很明显的局限性。比如它不支持非常精确毫秒级的处理。抛开这个限制，我希望有一种数据产品既可以充当 Kafka 的角色，也能支持流式数据进来。我可以编写简单的 SQL 处理函数在上面做一些计算，经过 Pipeline 可以支持很多函数，同时这些数据又可以再回到数据库里去消化掉，这个数据库同时又可以

外提供查询或者其他的能力。其实我想探讨的就是这个思维，all in one，我们需要一个叫做 Streaming Database 的概念，希望有一个流式数据库，把数据的 in 和 process，还有数据的 output 全部融在一起，让整个编程模型变得简单高效。相比之下之前所提的三件套就太过于标准化，过于陈旧，虽然它很吸引人，很优秀。

### 3. Elasticsearch + X 畅谈



我希望 ES 可以支持这三种门类，其实它已经支持了两种，就是数据的 process 里有 in, transformer, 还有 aggregation, 还有一些其他的函数能力。如果能把上游的流式解决，我相信未来它应该可以占领更多的流式的处理市场。

其实 ES 本身是支持索引的，如果把这个索引充当一个 Buffer 流就会面临，数据可以写进来但是读数据就比较麻烦，因为它的数据在顺序上没有得到很好的控制，数据可以从头写到尾，但我们无法像 Kafka 一样从尾读到头，这就是 ES 还需要改进的地方。所以希望未来市场上有更多 Streaming 产品，因为现在它已经具备了 Bash 批处理模拟，如果它未来加入了，我相信在 ES 的市场上绝对能够很好地发展。比如把 Kafka 的节点融进来；把 Kafka 的特性、具备的能力加入到 ES 里面来；创建索引的时候直接就可以创建一个队列的索引；写数据的时候可以从头写，读的时候可以从底部读，不用按照标准的索引 search 去查了。然后经过 process 节点的时候，processor 具备了 Flink 的处理能力和逻辑能力，最终输出到索引。

这就是我想和大家探讨的一个全新思维，虽然标题讲的是基于窗口的，但其实窗口计算在整个大数据里只是占了很小的一点，从 high level 这个层次来看，实时计算流法还是本质。

# Elasticsearch 在清博大数据的应用与实践

分享人：清博大数据技术副总裁王欢

视频地址：<http://cloud.video.taobao.com/play/u/3177173649/p/1/e/6/t/1/297145637513.mp4>

关于 Elasticsearch 在清博的应用和实践，本文将通过四个部分展开介绍：

- 关于清博
- 典型业务场景
- 痛点与诉求
- 阿里云 Elasticsearch 实践与收益

## 一、关于清博大数据

清博数据成立于 2014 年，是一家专注于内容数据领域，是基于网络公开数据，依托自然语言处理、知识图谱、事理图谱等 AI 技术，挖掘有价值的数据以及关系，为政务部门、企业、媒体和高校的网络空间数据治理，提供一站式解决方案的公司。

清博大数据的主要业务布局有：





- 清博指数，已经成为业内知名的新媒体账号评估服务体系；
- 清博舆情，是业内主流舆情 SaaS 服务平台，为数十万 B 端网络品牌跟踪管理提供一站式服务；
- 清博融媒，助力政企、校园的融媒体建设；
- 产业数据，基于数据和技术的积累，在汽车大数据和环保大数据等产业数据领域也有布局。

## 二、典型业务场景

上文所提到的业务方向中都有 Elasticsearch 的参与和支持。下面介绍几个典型的应用场景。

### 1. 清博舆情

下图展示的是清博舆情中常见的分析模块。



清博舆情利用 NLP 技术给常见的每条新闻及评论打上 7 个情绪标签，然后基于标签分析每天的情绪走势和情绪分布情况。这个应用可以理解为，针对某个事件，大众在不同情感表达下的声量大小。



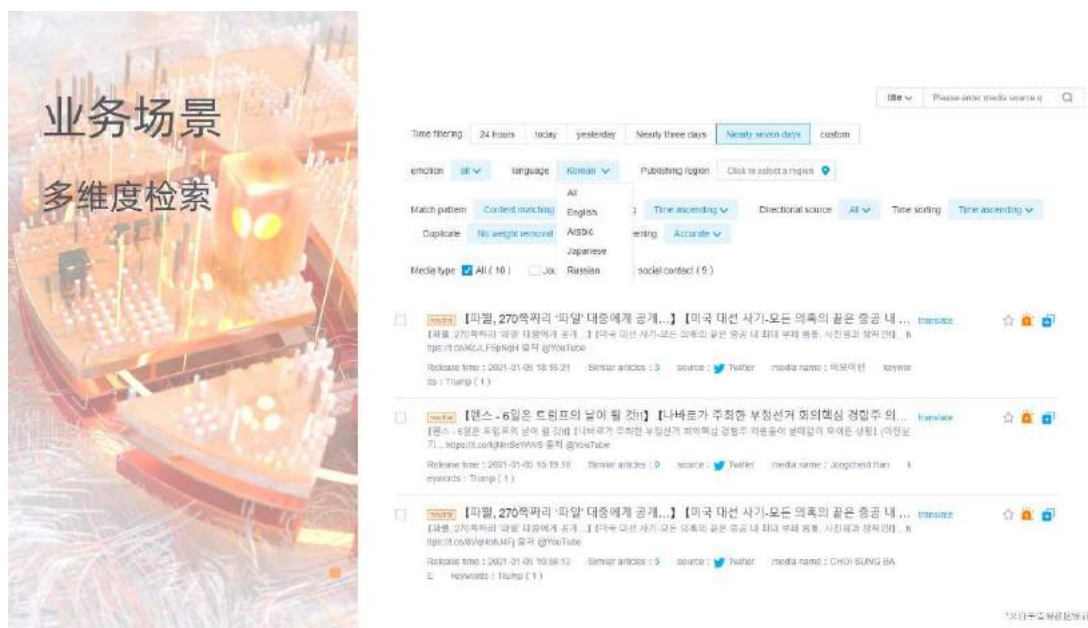
另外，我们也对事件的发布和媒体敏感信息报道进行渠道统计，统计后就可以直观的看到一个事件的重要或敏感的传播节点，这就为业务的决策者提供了非常直观的决策参考。除此之外，系统还为用户提供发文类型、提及地区、热词分析等十多个常用的统计分析模块。

以上所描述的统计分析，都是基于 Elasticsearch 强大的聚合统计能力，包括嵌套的统计能力实现的。由于清博大数据提供的是线上的 SaaS 服务，在同个页面会迸发请求查询或聚合多个接口等需求，这就对 Elasticsearch 的聚合统计性能和内存使用率，都提出了更高的要求。

## 2. 多维度的检索

多维度检索在清博大数据也是比较常见的业务场景。在舆情高级检索模块，清博检索提供了基于时间、发布平台、情感、媒体类别、发布地区等十多个维度的复合检索。同时还提供了基于 term 和 match phrase 的精确或模糊检索的复杂查询方式。

不同于日志检索场景，清博的多维度检索业务，需要权衡召回率和准确率，需要对标题跟正文设置不同的打分策略。比如给标题设置更高的权重，给正文设置相对较低的权重。另外，产品还能对包括阿拉伯语、日语、韩语、德语、法语等多国语言进行文本检索，这就涉及到了多语种的分词与检索召回。



3. 账号画像

下图是清博指数平台一个微信公众号的画像页面，展示了公众号系列数据，包括阅读统计、阅读点赞、在看、发文趋势、发布习惯等数据指标的统计模块。通过这些模块，用户可以直观的了解一个公众号的发布情况、文章传播情况等，不仅给运营者提供了全方位、可参考的数据，也给广告投放的用户提供客观投放参考。

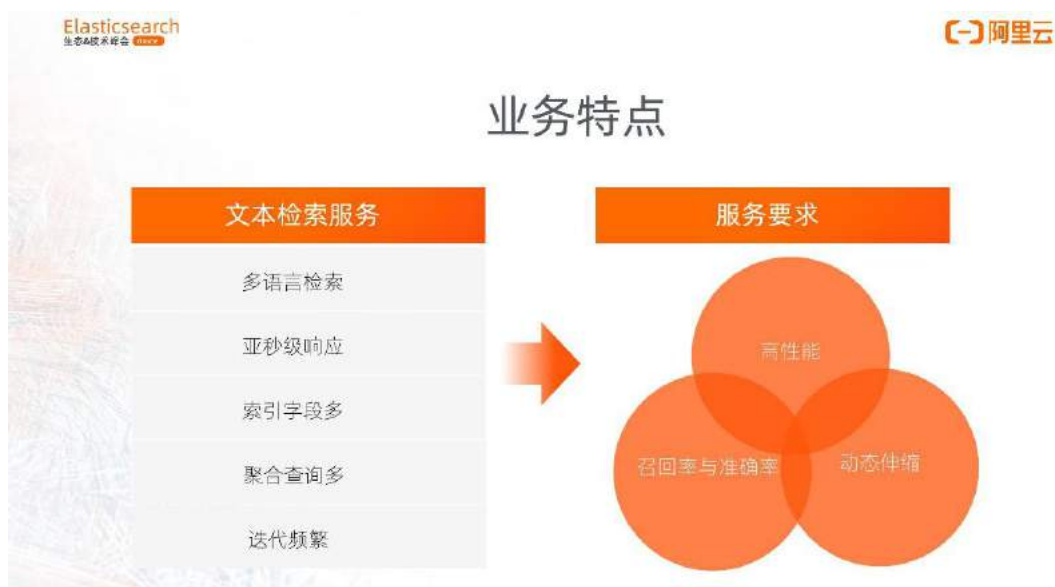


三、痛点与诉求

上文列举的不同业务场景都离不开 Elasticsearch 强大的查询和聚合能力，那么实现这样的功能有哪些痛点与诉求呢？

从上面业务场景列举可以看出，清博大数据的业务基本上都是基于文本检索，那么它具有的特点包括：

- 需要支持多种语言检索；
- 由于是在线的 SaaS 系统，需要实现亚秒级响应；
- 需要支持上百个索引字段；
- 需要满足聚合查询多和迭代频繁；
- 基于以上的业务特点就要求 Elasticsearch 具备高性能、平衡检索的召回率和准确率，并具有动态伸缩的能力。



基于以上的业务特点和服务要求，对清博大数据来讲，业务痛点可以分为三个方面：

### 1) 成本

单篇文档不同于日志数据，占用索引空间大，所需的 SSD 存储昂贵；

### 2) 运维

当业务数据不断增加，需要频繁扩节点；同时，由于是在线 SaaS 业务，所以需要实现亚秒级响应速度；并且在节点升级、增加内存等在线升级时，不停止服务。



### 3) 兼容性

实现 Elasticsearch 集群迁移、本地化部署，业务代码无感知适配；云端备份索引可以兼容开源的 Elasticsearch。

针对以上的业务痛点，清博大数据对云端 Elasticsearch 也提出了三点诉求：

#### 1) 成本

平滑扩缩容，索引支持冷热分离，热数据才放到 SSD 存储，从而降低存储成本；

#### 2) 运维

需要有简单的运维工具支持一键扩缩容和节点升级等能力；需要支持丰富的运维指标，在达到一定阈值时，可以有不同类型的报警通知；需要支持平滑升级不对业务产生影响；

#### 3) 兼容性

阿里云 Elasticsearch 100%兼容开源生态，并支持自定义的分词器，实现热更新分词词典，方便备份与恢复。云上备份与索引，可以快速在开源 Elasticsearch 上恢复或拉起服务。



## 四、云上实践与收益

清博大数据在迁移之前是基于 Apache Solr 自建的搜索集群，为了提升搜索性能和巩固集群稳定性，创建了很多小的 Solr 集群。在上游数据分析之后，通过一套路由机制写入

到不同的 Solr 集群，在查询时通过同一套机制，自动选择对应的集群。在不同的集群中查询不同的数据，可以分散单个集群压力，如果有小集群异常只会影响部分业务。

但是太多分散且小的集群也会出现很多问题，比如增加故障的概率、缺少统一运维管理平台、运维管理成本过高、无法自动扩充 Shard、需要手动增加节点扩容等等。



相比之下，阿里云 Elasticsearch 提供智能的、统一的运维平台，不仅可以多维度地监控告警，也能及时发现集群问题，而且阿里云 ES 专家的支持也减少了大量的运维成本。

阿里云 ES 自动 Shard，提升系统性能，提高了系统的检索性能；阿里云 ES 节点的伸缩能力，可以灵活应对业务逐步增加，节省大量一次性投入的成本。

在迁移过程中，清博大数据同步升级了数据架构，整个平台是分层的数据模式。最底层是数据接入层，包括上述提到的各平台社交文本数据。所有的数据源首先会推送到 Kafka 集群，通过消息队列对各个业务模块进行检索。

再往上一层是数据处理和存储层。在存储层，像新闻标题、发布时间、原始 URL 等信息存储到 HBASE。一些不需要检索内容的镜像数据会存到 OSS 上进行长期存储，并把 OSS 的路径存到 HBASE，方便后面的检查。

在数据处理这一层，使用 Flink 实时流处理引擎，通过 RPC 的方式实时调用 NLP 相关的算法服务，为每一条文本内容打上情感属性、情绪分类、新闻类型和地域等标签，方便下游业务使用。





再上一层是整个 ES 的基础设施，包括 ES 集群和智能网关两层服务。在 ES 集群层，根据业务特点把近两年的数据放到冷数据集群，使用价格相对较低的高效云盘，把近三个月的数据放到热数据集群，使用 SSD 盘进行冷热集群分离，同时根据不同平台大小对索引进行拆分，分为微信、微博、短视频等等索引。

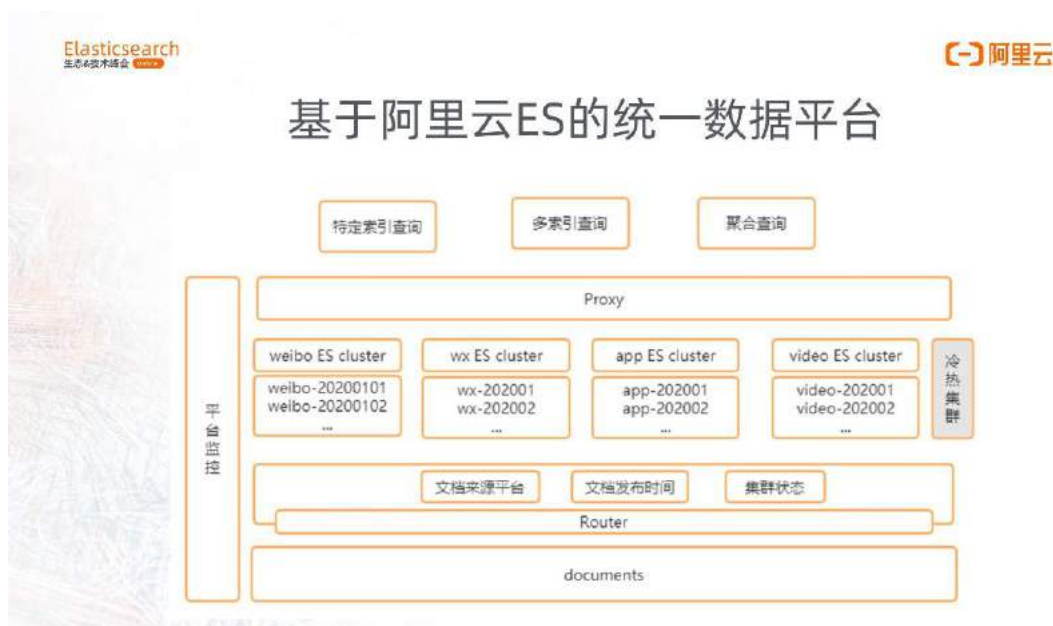
为了提高 ES 的安全性和访问的可控性，清博大数据开发了智能网关服务，屏蔽了直连 ES 的方式。在智能网关层，有很多业务都要调用 ES 数据，为了防止单个业务占用过多资源，影响到其他业务使用，网关可以为每个业务分配各自的 QPS 并设置优先级。一旦集群出现问题，可以对低优先级的业务进行熔断限流，以保证高优先级的业务。

由于在 ES 的数据层把索引按照来源进行拆分，数据被分散到多个集群中，这会给查询带来一定的麻烦。为了方便前端业务的调用，网关路由所有模块可以根据业务端查询的数据类型，自动路由到对应的索引，同时网关层也会对业务端查询进行优化，比如根据时间段选择对应索引，而不是扫描全部索引。

最上面一层是业务接入层，包括上文提到的舆情业务、指数业务、融媒业务等等。

基于阿里云 ES 打造的统一数据平台，也可以看作是一个分层架构，如下图。





最底层文档写入 ES 索引前, 会先进入路由层。路由服务会根据文档的来源、发布时间、机型状态等, 选择对应的集群以及索引, 比如自动选择对应平台或者是对应索引所在的机器。

对日增数据量比较少的平台, 会按照周或月进行索引拆分。对日增数据量很大的平台, 路由会按天创建索引, 这样就避免单个索引过大、数据过于倾斜而影响整个集群性能的问题, 使每个索引的大小保持基本相同。

再向上一层是多个 ES 集群, 包括不同平台的数据集群和冷热数据集群。前面一层经过路由策略之后, 最终数据会分散到对应集群的对应索引中。为了方便查询, 通过开发的 Proxy, 对用户特定的索引查询、多索引查询和聚合查询, 甚至跨集群查询等, 可以对数据进行查询集合, 做到对业务端底层的索引细节屏蔽、业务端无感知、降低业务端调用成本, 同时也方便底层的迭代升级。

基于以上的架构, 在阿里云的 ES 数据平台提供近三个月超过 100 亿的热数据, 在近三个月的索引占用空间超过 60TB, 日增网络公开内容数据超过 1.2 亿, 单篇文档搜索字段超过 200。



在这么大数据量的情况下，频繁变动升级会带来一些问题。值得一提的是，利用阿里云ES 诊断功能，可以很方便地发现 ES 集群潜在的问题，也为集群的运维方向提供诊断经营。

在阿里云 ES 各种功能的加持下，清博大数据平台的稳定性较之以前提升了 60%，整个运维时间减少了 80%。正是由于阿里云 Elasticsearch 平台的各种能力，让团队可以把更多的精力放到产品开发和迭代上，比如基于平台能力快速开发了轻薄融媒平台，为清博融媒平台提供内容检索服务。

在开发的政府补贴类应用中提供政策搜索服务，为后面的推荐算法提供出色的结果，也为年终公众号运营画像类应用提供相关统计服务。除此之外，基于平台的能力还能提供更多的产品形态。

# Elasticsearch 在企查查的应用实践

分享人：企查查搜索部门经理范兆明

视频地址：<http://cloud.video.taobao.com/play/u/3177173649/p/1/e/6/t/1/297441406466.mp4>

众所周知,企查查是一家专业做工商查询的公司,所有的业务入口都是基于查询完成的,可以说查询是实现企查查价值的主要入口。所以,本篇内容将介绍 Elasticsearch 在企查查的应用实践。

企查查遇到阿里云 ES 时的状况是,海量的数据无法存储,存储后的数据无法做大规模的分析,实时的用户行为得不到监控。基于这些痛点,让企查查在寻找解决办法的过程中遇到并认识了阿里云 ES。通过搭建 ELK 日志分析平台、日志分析、全文检索等功能,充分的了解和熟悉了 ES。

ES 主要的技术革新和特点,总结而言有三个:架构天生分布式、检索全文和结构、分析实时聚合。

## 一、架构天生分布式

ES 天生的分布式架构可以通过硬件扩容的方式实现海量数据的膨胀,并且它的副本模式能够解决数据安全问题。



## 二、检索全文和结构

通过 Lucene 的倒排索引、Bm25 的全文检索和高效的结构化检索，能够满足大部分搜索场景。

## 三、分析实时聚合

实时的海量聚合能力和多聚合模式能够完成大部分分析场景。

基于以上 ES 的能力，最终企查查选择了阿里云 ES。那么阿里云在 ES 的基础上又带来了哪些方面的便捷呢？



### 阿里云Elasticsearch带来的便捷



## 四、优秀的 NLP 分词器

阿里云 ES 具有优秀的 NLP 分词器能力。目前主流的 ES 中文分词器有 IK 和 ANSJ 等主要的几种，企查查选择了 IK 和 ANSJ 后，清洗了大约百万级的基础数据，然后导入到 IK 和 ANSJ。通过阿里云分词器对比了 IK 和 ANSJ 分词器发现，基于 NLP 的阿里云分词器更优秀。

区别就在于，基于 NLP 的阿里云分词器可以在不同场景解析出不同的语义，完成不同分词，所以企查查最终的解决方案是，以阿里云分词器为主，以 ANSJ 分词器为辅做了两套分词模式，同时应用于文档搜索。

## 五、一站式管理和高效扩展

这两个特点原本就是云平台的天然优势,这也是为什么中小企业和高速发展企业会选择云平台的根本原因。

企查查基于阿里云 Elasticsearch 设计的应用实践,这里给大家介绍几个核心的数据。



## 六、5000+QPS

5000+QPS 是指实时峰值 QPS 达到每秒 5000;

## 七、200+应用

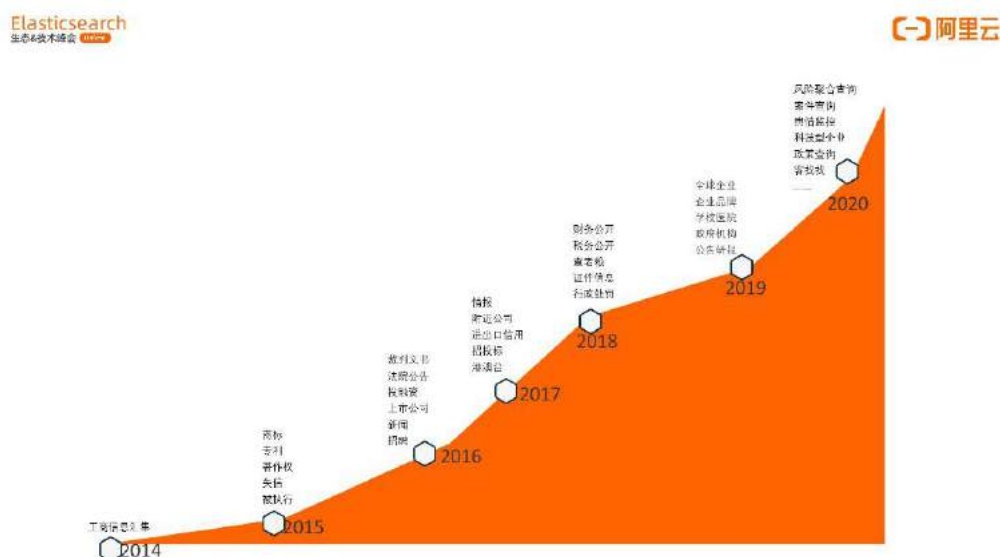
200+是指目前有 200 个数据维度参与了实时搜索;

## 八、8TB 数据

8TB 是指所有实时搜索数据加起来超过 8TB。

从下图企查查的发展历程可以发现,2014 年企查查只有单一的工商搜索维度;发展到 2020 年底,企查查已经超过了 200 多个搜索维度。快速的版本迭代、数据爆炸式的增长、

爆炸式的用户请求等等都在阿里云 ES 上都得到了很好的体现。因为阿里云具有足够稳定、快速扩容、大大减少运营成本和搜索故障等特点，在这些年的发展中不断的提升企查查的搜索体验。



## 九、企查查的技术特点

企查查涉及到的其他技术特点跟全文检索特点是一样的，包括高并发、海量数据、实时聚合和分词等特点。高并发和实时聚合也是 ES 所擅长的，再加上阿里云优秀的 NLP 分词器，两两作用后让企查查搜索体验更优秀。



## 十、搜索的目的



我们越来越重视搜索并不断提高搜索体验，那么搜索需要呈现的结果是怎样的呢？



第一个是精准搜索。顾名思义，就是把搜索词和文本词完全匹配的结果返回给用户；

第二个是分词匹配。分词匹配是目前搜索技术的主流，也是最难实现的。难度在于，虽然有基于语义的分词器，如基于 NLP 的阿里云，但仍然会有分词异常和分词歧义的时候。基础数据的权重配比、清洗排序能否达到用户预期和搜索意图，都是长期迭代的任务。

第三个是意义搜索。意思是当用户搜索的词，在精准搜索和分词搜索都没有匹配，但实际上意义是一样时提供的服务，比如番茄和西红柿，或是同音字和形近字，这个时候就会需要意义搜索。意义搜索就是把同义字、同音字和形近字都反馈给搜索用户。

第四个是意图搜索。当用户搜了一大段内容，但是在分词、精准和意义搜索里都没有办法找到搜索结果时，就需要提取用户搜索的核心词，然后用核心词再去搜索并反馈给用户结果。

第五个是部分匹配搜索。意思是当上述四个搜索都没有结果的时候，需要将部分匹配出的结果反馈给用户。

## 十一、Elasticsearch 还能带来什么？

目前 ES 在机器学习方面做了持续迭代，可以自动发现实时数据异常，自动实现业务的实时监控。

ES 可以做更深层次的数据分析，还可以将发现数据的核心词和数据的特点推荐给搜索的用户。

# Elasticsearch 在乐言的多场景应用实践

分享人：上海乐言信息科技有限公司 Elasticsearch、Kafka 负责人徐二涛

视频地址：<http://cloud.video.taobao.com/play/u/3177173649/p/1/e/6/t/1/296822688713.mp4>

关于 Elasticsearch 在乐言科技的多场景应用实践，本文将通过四个部分展开介绍：

- 背景介绍
- ES 的核心价值
- ES 在乐言的应用场景
- ES 面临的挑战

## 一、背景介绍

上海乐言信息科技有限公司（下文简称“乐言科技”）成立于 2016 年 4 月，致力于将 AI 前沿技术赋能各垂直行业客户，在电商客服、金融咨询、政务问答医疗问诊等领域有广泛应用。

乐言科技的 AI 智能客服产品“乐语助人”，具备了自动应答训、训练客服、协同人机交互等功能。

## 二、ES 的核心价值

展开介绍 ES 的核心价值前，首先介绍下 ES 在乐言科技的业务体量，Elasticsearch 在乐言科技支持了多条业务线，包括乐语助人、智能营销、智能推荐、汇乐读。

乐言科技各个业务使用 ES 的情况都不相同，目前共有 30 多套 ES 集群，每天产生 50T 的数据量，每月的存储数据量级达到 P 级，双十一期间每天写入数据甚至达到每秒 200 万。

ES 的核心价值离不开 Elastic Stack 生态圈。近几年 Elastic 有很大的发展，一方面它的使用场景很丰富，主要是基于 ES 的搜索和分析，具有多种场景应用。目前，ES 可以提供的解决方案有搜索、日志分析、指标分析和安全分析等。



在开源领域，ES 有比较成熟的技术组件，比如可视化的 Kibana，搜索和查询可以在 Kibana 汇集成各种各样的直观的图表和柱状图。Elasticsearch 是分布式存储计算一组多副本的，在保证高可用性下又保证性能。数据抓取主要有 Logstash 和 Beat。Beat 可以结合开源组件自己编写。

从商业的角度来看，X-pack 具有更丰富的功能，比如安全、报警、监控和图查询、机器学习等方面。这些 X-pack 的功能对于开源来说是比较缺乏的，当下很多云厂商都对以上这些功能做了封装。

### 三、ES 在乐言的应用场景

ES 在乐言科技的应用场景很丰富，比如日志分析搜索、业务搜索、运营数据、数据分析等等。

在日志系统，乐言科技主要收集外部日志、系统日志、堡垒机日志和业务日志。在业务

搜索领域，主要包括商家后台的话术搜索、问答搜索和商品过滤。运营数据包括运营系统的

Galaxy 系统、运营人员和图标绘制。Galaxy 系统也是基于 ES 研发的，由于 ES 开箱即用的搜索功能和数据分析功能，大大提高了开发效率。



## ES在乐言的应用场景



在运营系统中，运营人员使用 Galaxy 系统可以很好的进行数据分析，在 Kibana 上也可以进行多种多样的图标绘制。数据分析后，Kibana 也可以支持导出。另外，还有丰富的 Rest API 可以调用。

在数据分析方面，乐言科技可以对用户的数据打各种各样的标签，然后每天离线实时写入 ES，最后生成用户画像、用户标签和统计分析。

总而言之，在数据挖掘和分析方面，ES 提供了不可小觑的能力。

### 1. ES 在乐言科技的应用场景 —— 日志搜索分析

日志系统在乐言科技有很广泛的应用。乐言科技的日志架构是经典的 ELK 架构。ELK 架构最左边是系统日志，通过 syslog 输出到 Kafka；Web 日志通过 Logstash 做一些解析，然后将结构化的数据写入到 Kafka；容器日志是基于 filebeats 实现服务发现的一些功能，可以将容器打上不同的 tag，再将需要收集的容器打到不同的 Kafka 的 Topic；应用日志封装了 SDK，通过 log 组件接入 Kafka-appender，然后将数据通过序列化再写入 Kafka。

Kafka 基于 goagent 到 Elasticsearch，不仅支持基于 pro buff 的解析，针对 logstash 来说也能节省不少资源，大大提高 Elasticsearch 的写入。Kibana 是基于业务

日志和系统日志的搜索。

91

> Elasticsearch 在乐言的多场景应用实践

在日志采集中，应用日志主要用于异常日志和慢日志，用来定位问题；Web 日志主要用于监控 web 服务器的运营情况，以及一些 Web 服务器请求情况的监控；系统和堡垒机日志，可以进行一些日志的分析，用来做安全分析。

Elasticsearch  
生态技术峰会

阿里云

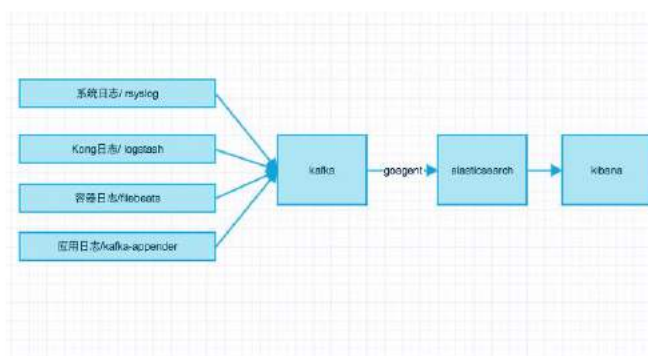
## ES在乐言的应用场景

### 日志采集：

- 应用日志
- kong日志
- 系统日志
- 容器日志

### 主要特点：

- 完整的解决方案，易于运维
- 实时性：从日志产生到访问，10s级
- 交互式分析：亿级别日志，搜索秒级响应
- 全文搜索：基于倒排索引，搜索分析灵活



ES 为日志提供了一套比较完美的解决方案，它的主要特点就是任何一个开发运维的同学都可以使用成熟组件，并通过简单的部署搭建一个完整的日志分析服务。从日志的实时性来讲，即使在亿级的日志也能达到 10 秒级的查询，相对于传统的数据解决方案小时级的时效性都是非常高的。即使在亿级别的日志，在交互式分析上 ES 的搜索响应也可以做到秒级。

ES 提供了非常丰富的查询分析能力，支持全文搜索。由于 ES 基于倒排索引和存储数据结构，基本所有不同类型的搜索都可以实现。作为不同行业日志搜索最广泛的形式，ES 非常完美地解决了日志实时分析应用场景，这也是最近几年 ES 快速发展的原因之一。

### ES 在乐言科技的应用场景 —— Web 日志统计分析

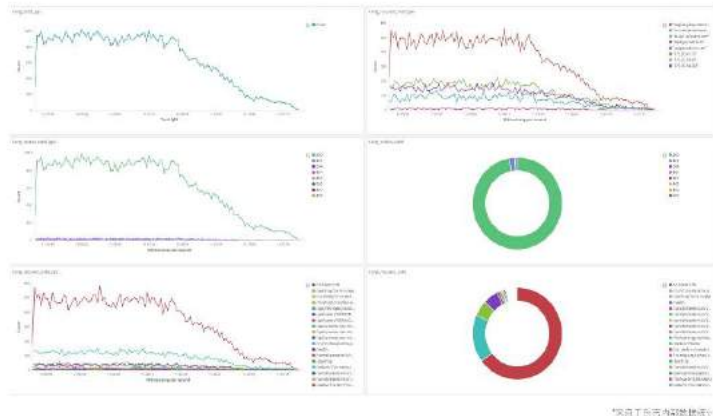
针对 Web 的请求监控，ES 可以对请求状态码占比及实时的 qps 进行绘制图标，比如基于 timeline 走势的绘图，或基于一些维度的饼状图等等。



## ES在乐言的应用场景

### Web日志统计分析:

- web请求监控
- 请求状态码占比及实时qps
- request\_path的占比及实时qps
- 交互式的可视化排障
- GeoIP



在这样的 Web 日志系统，排障是交互式的可视化操作，非常方便。比如发现一个请求有大量的 502，就可以在 Kibana 上通过过滤条件过滤出 502 的相关请求，可以看到 502 的 surname 等信息。总而言之，在 Web 日志分析方面有很强的竞争力。

另外，ES 还可以支持扩展，可将来源 IP 进行 GeoIP 库的解析，比如解析成地理位置，或查看 Web 请求分布的情况。

## 2. ES 在乐言科技的应用场景 —— 业务搜索

业务搜索在商家后台，包括商品过滤、话术搜索、问答搜索和搜索结果排序等功能。对 ES 搜索条件可以进行多维度的组合。

在业务搜索方面来说，最主要的特点是高可用性，ES 的高可用性可达到 99.99%。这依赖于阿里云 ES，可以创建跨区的部署，支持单机可用的故障容灾。业务搜索一般都比较复杂，ES 搜索的高性能也非常突出。它支持单台万级 qps 搜索，响应实践平均在 30 毫秒以内。

## 3. ES 在乐言科技的应用场景 —— 运营数据分析

运营数据分析可以对店铺、订单、商品或用户等维度绘制运营数据监控图表，包括饼图、



柱状图或基于时间线的线状图。

## ES在乐言的应用场景

### 运营数据分析：

- 运营数据图表绘制
- 多纬度聚合、排序，支持导出数据
- 运营人员学习成本低



\*此图并非真实数据设计

由于 ES 在多维度的聚合排序是相当强劲的，所以 ES 在开发能力上有丰富的 Rest API 非常方便，对于开发效率的提升也很有帮助。

ES 有成熟的可视化工具，这就对运营成员很友好，使得他们的学习成本降低，非常简单的就可以绘制各种分析或监测所需的图表，包括在 Kibana 中检索的数据做过滤，或用于其他的分析等等。

## 4. ES 在乐言科技的应用场景 —— 数据分析

乐言科技的数据分析会给用户打多种多样的标签，离线统计之后形成用户画像，写入 ES，再通过 ES 的强大聚合和分析能力，观察老用户的流失率和新用户的情况，包括复购用户等等很多有价值的数据。同时，基于 ES 搜索分析的能力，在大数据报表生成方面，也是很方便的。

总而言之，ES 在未来数据挖掘中可以提供更多帮助。

## 四、ES 面临的挑战

ES 在使用过程中面临的挑战包括日志和搜索两方面。

## 1. 日志的挑战

从日志的角度来看，数据量非常大，每天有超过 50T 的数据产生，若保留 30 天就会达到 P 级的日志；日志还有低延迟和高并发写入的特点，这就需要 SSD 存储，随之而来的就是成本高的问题。

所以从日志的角度来看，挑战就是还没有很好的节约成本的架构。乐言科技的解决方案是采取 hot&warm+oss 归档的架构。Hot 主要是基于 SSD 节点进行存储，且只保留 7 天数据，超过 7 天会定期转存到 warm 节点，它的成本相对较低。同时这些数据都会写入 OSS 归档一份，用于查询超过 1 个月的数据。另外 Logstash 和 Beat 在 Kafka 写的效率不高，这里乐言科技重新编写了 goagent，以节省资源。

另外，通过 APP 名称建索引，然后不同 APP 名称对应的索引在 ES 中表留了时效性，也可以节省成本，包括约束研发、打印日志的规范等等。

## 2. 搜索的挑战

搜索的挑战主要是高可用性和高性能，不仅要达到 99.99% 可用性还要做到比较丰富的性能查询。所以从单机可用区故障容灾方面来讲，主要是使用阿里云 ES 在创建过程中创建的多可用区，即使这样也避免不了 ES 集群会有故障。所以在重要的 ES 搜索方面，乐言科技的解决方案是做了一主一备的搜索。一主一备是双写，主份是以查询为主的，备份也可以进行少量级别的查询。如果主份有问题，程序可以自动接入到备份集群。

在开源 Elasticsearch 的安全性也是很大的挑战。针对这个问题的解决方案，乐言科技做了基于 Aliyun Elasticsearch 支持的 X-pack 认证，通过不同的 role 和 user 完成关系架构，然后分配给所有级别一个权限，以保证数据的安全性。



钉钉扫码加入  
“Elasticsearch 技术交流群”



扫码了解  
“更多阿里云 Elasticsearch”



扫码订阅  
“阿里云 Elasticsearch 技术博客”



阿里云开发者“藏经阁”  
海量免费电子书下载