

- 全面讲解 Kibana 布局，面板操作说明，仪表盘定制方法
- 支持新版 Elasticsearch 聚合函数功能
- 完整的认证授权体系



三斗室 著

# Kibana 中文指南

NOT' REALLY®

*A committee - and look at the mess we ended up with*

# Table of Contents

---

- 1. 简介
- 2. Kibana 3 指南
  - i. 10 分钟入门
  - ii. 请求和过滤
  - iii. 行和面板
  - iv. 保存和加载
  - v. 仪表板纲要(schema)
  - vi. 模板和脚本
  - vii. 配置文件
  - viii. 面板
    - i. histogram
    - ii. table
    - iii. map
    - iv. bettermap
    - v. terms
    - vi. column
    - vii. stats
    - viii. query
    - ix. trends
    - x. text
    - xi. sparklines
    - xii. hits
    - xiii. goal
    - xiv. percentile
    - xv. range
  - ix. 认证鉴权
    - i. 用 nodejs 实现 CAS 认证
    - ii. 用 Perl 实现的认证鉴权框架
  - x. 源码剖析与二次开发
    - i. 源码目录结构
    - ii. 入口和模块依赖
    - iii. 控制器和服务
    - iv. 面板相关指令
    - v. 面板内部实现
- 3. Kibana 4 指南
  - i. 安装和运行
  - ii. 访问
  - iii. discover
  - iv. visualize
  - v. dashboard
  - vi. 配置
  - vii. 生产环境部署
  - viii. 新功能介绍
  - ix. 源码剖析
    - i. 主页入口
    - ii. 搜索页
    - iii. 可视化页
    - iv. 仪表板页
    - v. 设置页
    - vi. .kibana 索引数据结构
- 4. 捐赠名单

# 简介

---

Kibana 是一个使用 Apache 开源协议的 Elasticsearch 分析和搜索仪表板。已经历经了 v1, v2, v3, v4 个版本，分别采用了 PHP, Ruby, AngularJS, JRuby, NodeJS 等不同语言编写。本书主要介绍 v3 和 v4 的使用。

## 注释

---

本书原始内容来源[Elasticsearch 官方指南 Kibana 部分](#)，并对 v3 的 panel 部分加以截图注释。在有时间的前提下，将会添加更多关于 kibana 源码解析和第三方 panel 的介绍。

## 译作者的话

---

Kibana 因其丰富的图表类型和漂亮的前端界面，被很多人理解成一个统计工具。而我个人认为，ELK 这一套体系，不应该和 Hadoop 体系同质化。定期的离线报表，不是 Elasticsearch 专长所在(多花费分词、打分这些步骤在高负载压力环境下太奢侈了)，也不应该由 Kibana 来完成(每次刷新都是重新计算)。Kibana 的使用场景，应该集中在两方面：

- 实时监控

通过 histogram 面板，配合不同条件的多个 queries 可以对一个事件走很多个维度组合出不同的时间序列走势。时间序列数据是最常见的监控报警了。

- 问题分析

通过 Kibana 的交互式界面可以很快的将异常时间或者事件范围缩小到秒级别或者个位数。期望一个完美的系统可以给你自动找到问题原因并且解决是不现实的，能够让你三两下就从 TB 级的数据里看到关键数据以便做出判断就很棒了。这时候，一些非 histogram 的其他面板还可能会体现出你意想不到的价值。全局状态下看似很普通的结果，可能在你锁定某个范围的时候发生剧烈的反方向的变化，这时候你就能从这个维度去重点排查。而表格面板则最直观的显示出你最关心的字段，加上排序等功能。入库前字段切分好，对于排错分析真的至关重要。

以上是我在和同事就 ES 跟 Hadoop 对比的谈话中形成的思路。特此留笔。2014 年 8 月 28 日

---

关于 elk 的用途，我想还可以参照其对应的商业产品 splunk 的场景：

使用 Splunk 的意义在于使信息收集和处理智能化。而其操作智能化表现在：

1. 搜索，通过下钻数据排查问题，通过分析根本原因来解决问题；
2. 实时可见性，可以将对系统的检测和警报结合在一起，便于跟踪 SLA 和性能问题；
3. 历史分析，可以从中找出趋势和历史模式，行为基线和阈值，生成一致性报告。

——2014 年 11 月 17 日摘自 Peter Zadrozny, Raghu Kodali 著/唐宏，陈健译《Splunk 大数据分析》

## 参阅

---

- [Elasticsearch 权威指南](#)
- [精通 Elasticsearch](#)
- [Logstash 最佳实践](#)
- [The Logstash Book](#)

## 进度

---



欢迎捐赠，作者支付宝账号：[rao.chenlin@gmail.com](mailto:rao.chenlin@gmail.com)



# 简介

---

Kibana 是一个使用 Apache 开源协议，基于浏览器的 Elasticsearch 分析和搜索仪表板。Kibana 非常容易安装和使用。整个项目都是用 HTML 和 Javascript 写的，所以 Kibana 不需要任何服务器端组件，一个纯文本发布服务器就够了。Kibana 和 Elasticsearch 一样，力争成为极易上手，但同样灵活而强大的软件。

# 10 分钟入门

Kibana 对实时数据分析来说是特别适合的工具。本节内容首先让你快速入门，了解 Kibana 所能做的大部分事情。如果你还没下载 Kibana，点击右侧链接：[下载 Kibana](#)。我们建议你在开始本教程之前，先部署好一个干净的 Elasticsearch 进程。

到本节结束，你就会：

- 导入一些数据
- 尝试简单的仪表板
- 搜索你的数据
- 配置 Kibana 只显示你的新索引而不是全部索引

我们假设你已经：

- 在自己电脑上安装好了 Elasticsearch
- 在自己电脑上搭建好了网站服务器，并把 Kibana 发行包解压到了发布目录里
- 对 UNIX 命令行有一点了解，使用过 `curl`

## 导入数据

我们将使用莎士比亚全集作为我们的示例数据。要更好的使用 Kibana，你需要为自己的新索引应用一个映射集(mapping)。我们用下面这个映射集创建“莎士比亚全集”索引。实际数据的字段比这要多，但是我们只需要指定下面这些字段的映射就可以了。注意到我们设置了对 speaker 和 play\_name 不分析。原因会在稍后讲明。

在终端运行下面命令：

```
curl -XPUT http://localhost:9200/shakespeare -d '  
{  
  "mappings" : {  
    "_default_" : {  
      "properties" : {  
        "speaker" : {"type": "string", "index": "not_analyzed"},  
        "play_name" : {"type": "string", "index": "not_analyzed"},  
        "line_id" : { "type" : "integer" },  
        "speech_number" : { "type" : "integer" }  
      }  
    }  
  }  
};
```

很棒，我们这就创建好了索引。现在需要做的时导入数据。莎士比亚全集的内容我们已经整理成了 Elasticsearch 批量导入所需要的格式，你可以通过[shakespeare.json](#)下载。

用如下命令导入数据到你本地的 Elasticsearch 进程中。这可能需要一点时间，莎士比亚可是著作等身的大文豪！

```
curl -XPUT localhost:9200/_bulk --data-binary @shakespeare.json
```

## 访问 Kibana 界面

现在你数据在手，可以干点什么了。打开浏览器，访问已经发布了 Kibana 的本地服务器。

**Welcome to Kibana.**

Glad you could make it. Happy to have you here! Let's get started, shall we?

**Requirements**

- A good browser.  
The latest version of Chrome or Firefox is recommended. Safari (latest version) and Internet Explorer 9 and above are also supported.
- A webserver.  
Just somewhere to host the HTML and Javascript. Basically any webserver will work.
- Elasticsearch  
0.9.0 or above.

**Configuration**

If Kibana and Elasticsearch are on the same host, and you're using the default Elasticsearch port, then you're all set. Kibana is configured to use that setup by default!

If not, you need to edit config.js and set the elasticsearch parameter with the URL (including port, probably 9200) of your Elasticsearch server. The host part should be the entire, fully qualified domain name, or IP, **not localhost**.

**Are you a Logstash User?**

- YES - Great! We have a global dashboard: [\(Logstash Dashboard\)](#). See the note to the right about making it your global default.
- NO - Hey, no problem, you just have a bit of setup to do. You have a few choices:

1. [Sample Dashboard](#) I don't have much data yet, please extract some basics for me
2. [Unconfigured Dashboard](#) I have a lot of data and I don't want Kibana to query it at once
3. [Blank Dashboard](#) I'm comfortable figuring it out on my own

如果你解压路径无误(译者注：使用 github 源码的读者记住发布目录应该是 `kibana/src/` 里面)，你已经就可以看到上面这个可爱的欢迎页面。点击 Sample Dashboard 链接

**YOUR BASIC DASHBOARD**

**QUERY**

**Toggle query**

**FILTERING**

**HAVE A TIMESTAMP SOMEWHERE?**

If you have a field with a timestamp in it, you can set a time filter using the control in the navigation bar. You'll need to click the cog icon to configure the field that your timestamp is in.

**ABOUT FILTERS**

See the Filters bar above? Click it to expand the filters panel. Right now there are none. click on one of the icons in the document types list to filter down to only that document type

**DOCUMENT TYPES**

Term	Count	Action
line	110487	
scene	729	
act	180	
Missing field	0	
Other values	0	

**THE MOST GENERIC DASHBOARD EVER**

It's the best I can do without knowing much about your data! I've tried to pick some sane defaults for you. The two terms panels to the left of this text panel show a breakdown of your document type. Kibana is currently configured to point at the special Elasticsearch \_all index. You can change that by clicking on the cog icon in the navigation bar at the top. You can also add rows from that dialog. You can edit individual panels by click on the cog icon on the panel you want to edit.

The table panel below has attempted to list your fields to the left, select a few to view them in the table. To add more panels, of different types, click the cog on the row label to the far left.

**DOCUMENTS**

0 to 100 of 500 available for paging

**Fields**

All (1) / Current (8)  
Type to filter...  
 \_id  
 \_index

**\_source (select columns from the list to the left)**

```

["line_id":5964,"play_name":"Henry VI Part 1","speech_number":14,"line_number":54.60,"speaker":"JOAN LA PUCELLE","text_entry":"Then, Joan, discover thine infamy."}
["line_id":5947,"play_name":"Henry VI Part 1","speech_number":11,"line_number":54.53,"speaker":"JOAN LA PUCELLE","text_entry":"Will cry for vengeance at the gates of heaven."}
["line_id":5942,"play_name":"Henry VI Part 1","speech_number":11,"line_number":54.48,"speaker":"JOAN LA PUCELLE","text_entry":"To compass wonders but by help of devils."}
["line_id":5940,"play_name":"Henry VI Part 1","speech_number":11,"line_number":54.46,"speaker":"JOAN LA PUCELLE","text_entry":"Because you want the grace that others have."}

```

好了，现在显示的就是你的 sample dashboard！如果你是用新的.elasticsearch 进程开始本教程的，你会看到一个百分比占比很重的饼图。这里显示的是你的索引中，文档类型的情况。如你所见，99% 都是 lines，只有少量的 acts 和scenes。

再下面，你会看到一长段 JSON 格式的莎士比亚诗文。

## 第一次搜索

Kibana 允许使用者采用 Lucene Query String 语法搜索 Elasticsearch 中的数据。请求可以在页面顶部的请求输入框中书写。

Time filter • Q F A D S C

**FILTERING** •

**HAVE A TIMESTAMP SOMEWHERE?** • + X

If you have a field with a timestamp in it, you can set a time filter using the control in the navigation bar. You'll need to click the cog icon to configure the field that your timestamp is in.

**ABOUT FILTERS** • + X

See the Filters bar above? Click it to expand the filters panel. Right now there are none. Click on one of the icons in the document types list to filter down to only that document type.

在请求框中输入如下内容。然后查看表格中的前几行内容。

**DOCUMENTS** • + X

**Fields** •

All (1) / Current (9) •

Type to filter... •

- `_id`
- `_index`
- `_type`
- `line_id`

**\_source (select columns from the list to the left)** •

0 to 100 of 500 available for paging •

<code>line_id</code>	<code>play_name</code>	<code>speech_number</code>	<code>line_number</code>	<code>speaker</code>	<code>text_entry</code>
47871	Julius Caesar	30	3.2.82	ANTONY	Friends, Romans, countrymen, lend me your ears;
47800	Julius Caesar	6	3.2.14	BRUTUS	Romans, countrymen, and lovers! hear me for my
38512	Henry V	1	4.0.34	Chorus	And call them brothers, friends and countrymen.
96477	Titus Andronicus	2	1.1.9	BASSANIO	Romans, friends, followers, favorites of my right,
82971	Merchant of Venice	28	3.2.226	BASSANIO	I bid my very friends and countrymen,

关于搜索请求的语法，请阅读 [Queries and Filters](#)。

## 配置另一个索引

目前 Kibana 指向的是 Elasticsearch 一个特殊的索引叫 `_all`。`_all` 可以理解为全部索引的大集合。目前你只有一个索引，`shakespeare`，但未来你会有更多其他方面的索引，你肯定不希望 Kibana 在你只想搜《麦克白》里心爱的句子的时候还要搜索全部内容。

配置索引，点击右上角的配置按钮：

Time filter • Q F A D S C

**HAVE A TIMESTAMP SOMEWHERE?** • + X

If you have a field with a timestamp in it, you can set a time filter using the control in the navigation bar. You'll need to click the cog icon to configure the field that your timestamp is in.

**ABOUT FILTERS** • + X

See the Filters bar above? Click it to expand the filters panel. Right now there are none. Click on one of the icons in the document types list to filter down to only that document type.

在这里，你可以设置你的索引为 `shakespeare`，这样 Kibana 就只会搜索 `shakespeare` 索引的内容了。

**Dashboard Settings**

**General** **Index** **Rows** **Controls** **Timepicker**

**Index Settings**

**Timestamping** **Default Index** • **Preload Fields** •

`none` • `_all` •

**Save** **Cancel**

**DOCUMENT TYPES** • + X

**DOCUMENT TYPES** • + X

**THE MOST GENERIC DASHBOARD EVER** • + X

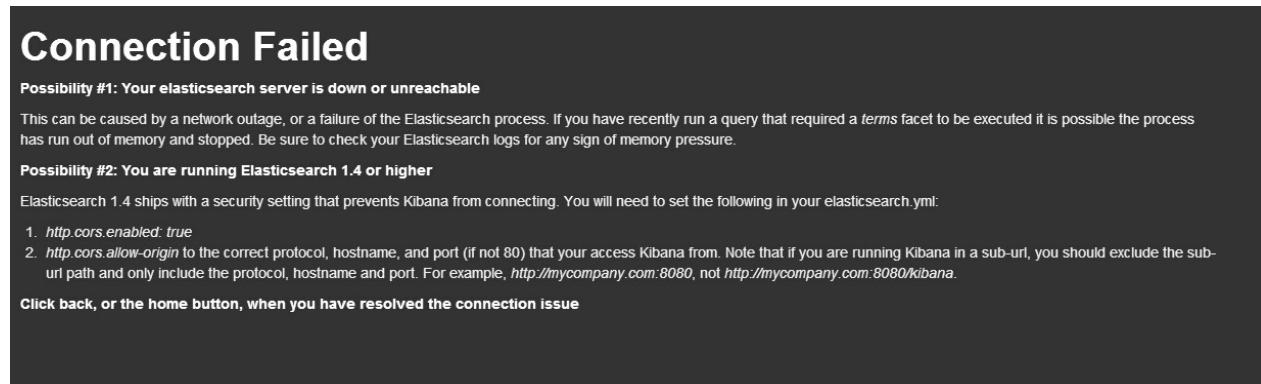
It's the best I can do without knowing much about your data! I've tried to pick some sane defaults for you. The two terms panels to the left of this text panel show a

下一步

恭喜你，你已经学会了安装和配置 Kibana，算是正式下水了！下一步，打开我们的视频和其他教程学习更高级的技能吧。现在，你可以尝试在一个空白仪表板上添加自己的面板。这方面的内容，请阅读 [Rows and Panels](#)。

## 译者注

在 Elasticsearch 发布 1.4 版后，使用 kibana3 访问 ES1.4 集群，会显示如下错误：



这是因为 ES1.4 增强了权限管理。你需要在 ES 配置文件 `elasticsearch.yml` 中添加下列配置并重启服务后才能正常访问：

```
http.cors.enabled: true  
http.cors.allow-origin: "*"
```

记住 kibana3 页面也要刷新缓存才行。

此外，如果你可以很明确自己 kibana 以外没有其他 http 访问，可以把 kibana 的网址写在 `http.cors.allow-origin` 参数的值中。比如：

```
http.cors.allow-origin: "/https?:\/\/kbndomain/"
```

# 请求和过滤

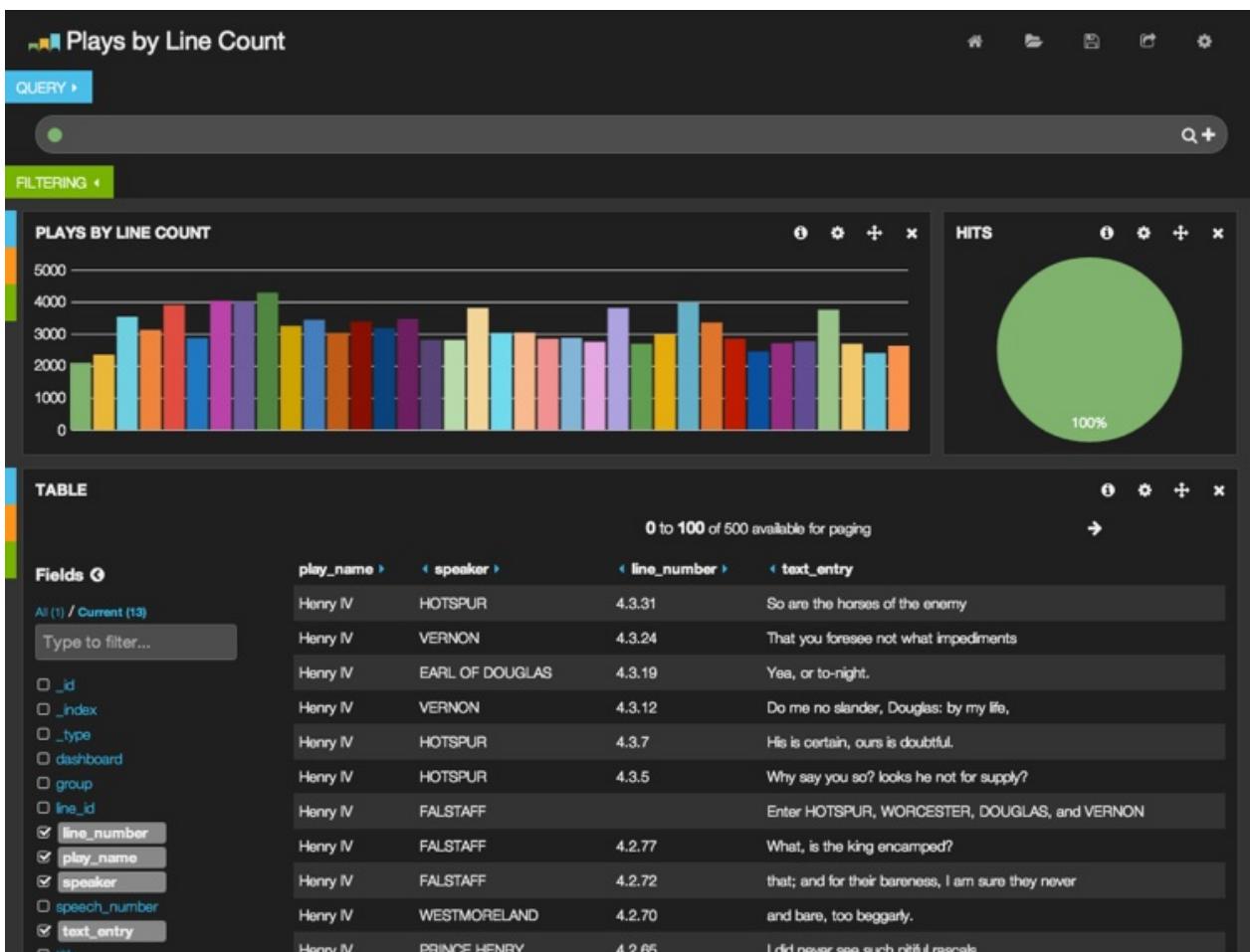
图啊，表啊，地图啊，Kibana 有好多种图表，我们怎么控制显示在这些图表上的数据呢？这就是请求和过滤起作用的地方。Kibana 是基于 Elasticsearch 的，所以支持强大的 Lucene Query String 语法，同样还能用上 Elasticsearch 的过滤器能力。

我们假设你已经：

- 在自己电脑上安装好了 Elasticsearch
- 在自己电脑上搭建好了网站服务器，并把 Kibana 发行包解压到了发布目录里
- 读过 [Using Kibana for the first time](#) 并且按照文章内容准备好了存有莎士比亚文集的索引

## 我们的仪表板

我们的仪表板像下面这样，可以搜索莎士比亚文集的内容。如果你喜欢本章截图的这种仪表板样式，你可以[下载导出的仪表板纲要\(dashboard schema\)](#)



## 请求

在搜索栏输入下面这个非常简单的请求

```
to be or not to be
```

你会注意到，表格里第一条就是你期望的《哈姆雷特》。不过下一行却是《第十二夜》的安德鲁爵士，这里可没有"to be"，也没有"not to be"。事实上，这里匹配上的是 `to OR be OR or OR not OR to OR be`。

我们需要这么搜索(译者注：即加双引号)来匹配整个短语：

```
"to be or not to be"
```

或者指明在某个特定的字段里搜索：

```
line_id:86169
```

我们可以用 AND/OR 来组合复杂的搜索，注意这两个单词必须大写：

```
food AND love
```

还有括号：

```
("played upon" OR "every man") AND stage
```

数值类型的数据可以直接搜索范围：

```
line_id:[30000 TO 80000] AND havoc
```

最后，当然是搜索所有：

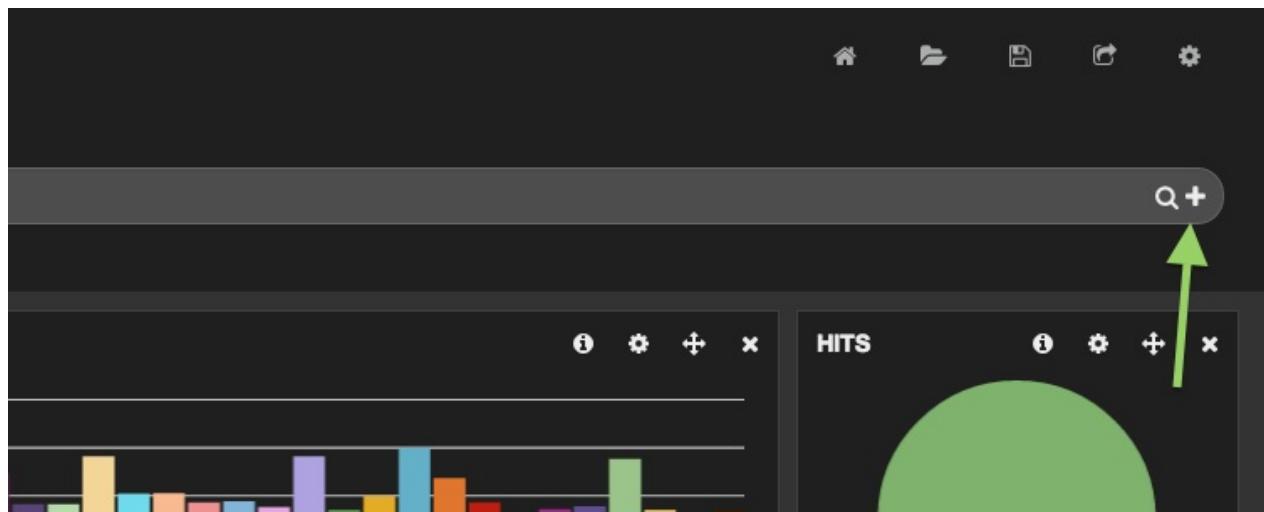
```
*
```

## 多个请求

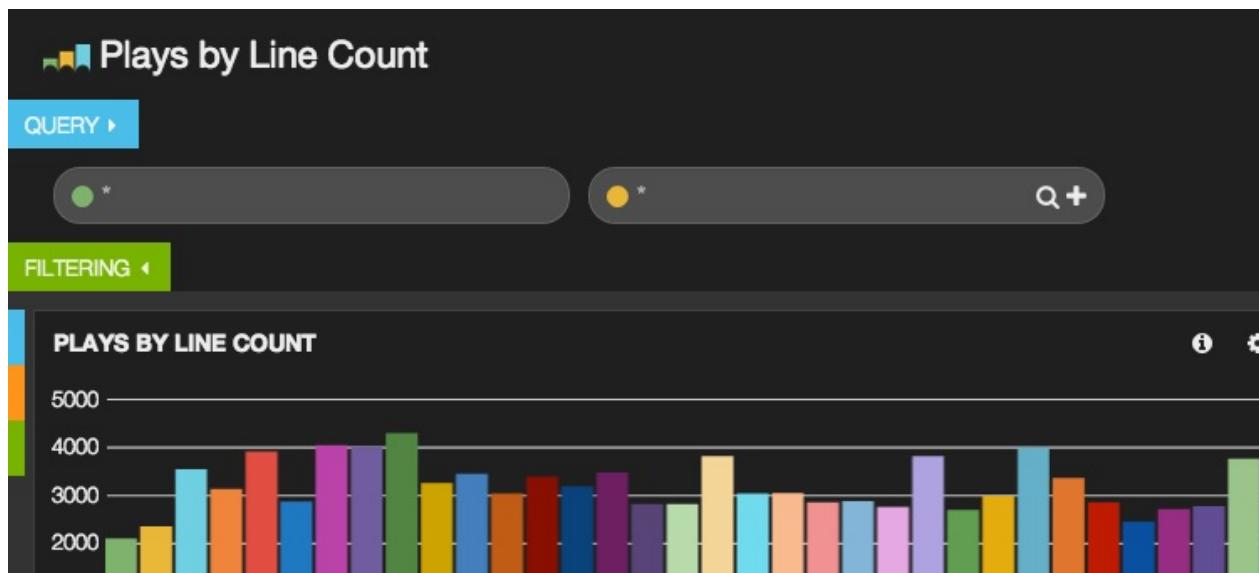
有些场景，你可能想要比对两个不同请求的结果。Kibana 可以通过 OR 的方式把多个请求连接起来，然后分别进行可视化处理。

添加请求

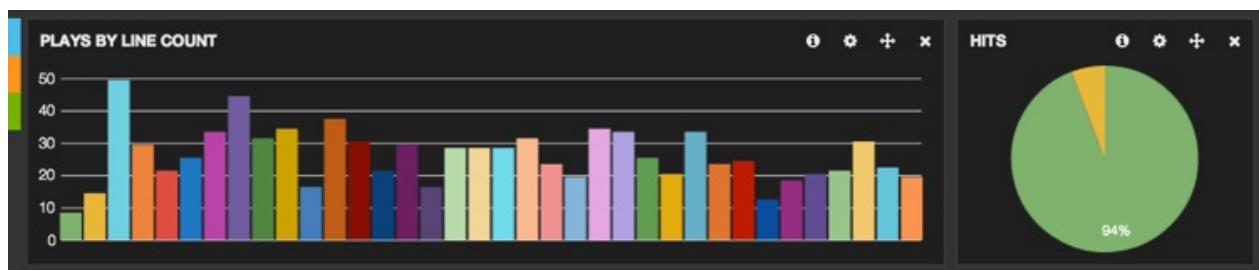
点击请求输入框右侧的 + 号，即可添加一个新的请求框。



点击完成后你应该看到的是这样子

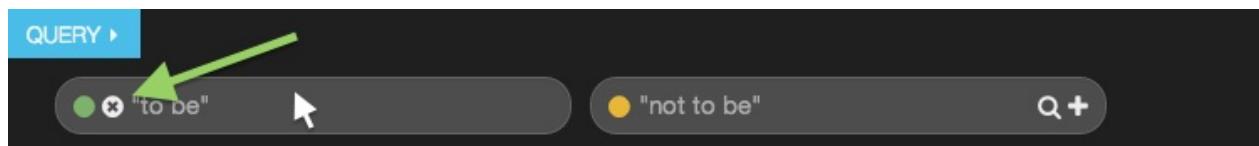


在左边，绿色输入框，输入 "to be" 然后右边，黄色输入框，输入 "not to be"。这就会搜索每个包含有 "to be" 或者 "not to be" 内容的文档，然后显示在我们的 hits 饼图上。我们可以看到原先一个大大的绿色圆形变成下面这样：



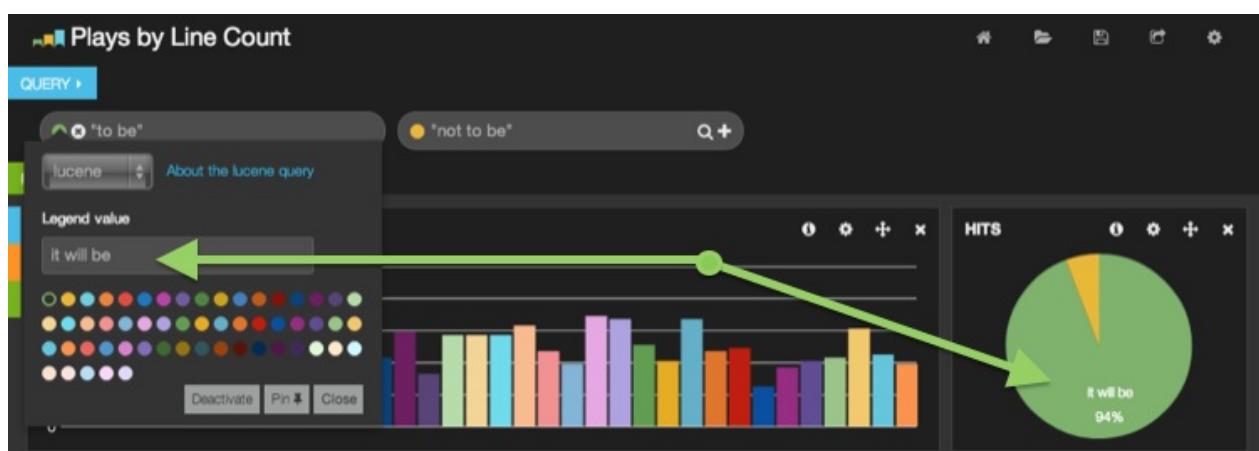
移除请求

要移除一个请求，移动鼠标到这个请求输入框上，然后会出现一个 x 小图标，点击小图标即可：



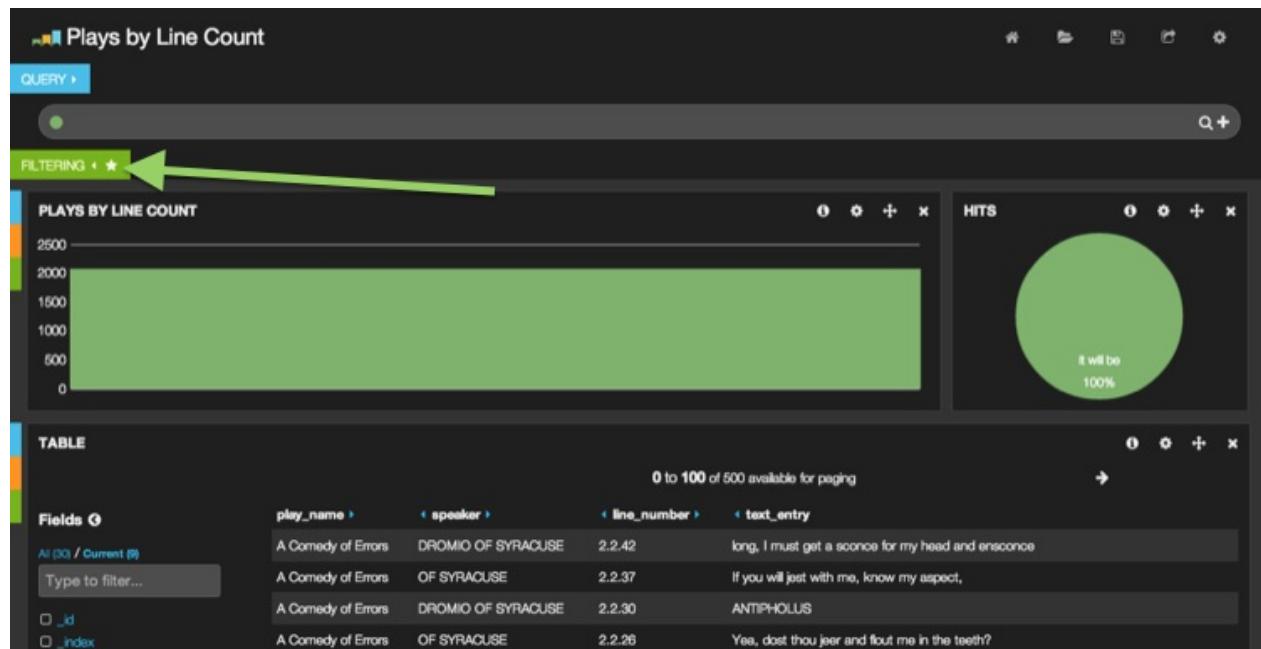
## 颜色和图例

Kibana 会自动给你的请求分配一个可用的颜色，不过你也可以手动设置颜色。点击请求框左侧的彩色圆点，就可以弹出请求设置下拉框。这里面可以修改请求的颜色，或者设置为这个请求设置一个新的图例文字：



## 过滤

很多 Kibana 图表都是交互式的，可以用来过滤你的数据视图。比如，点击你图表上的第一个条带，你会看到一些变动。整个图变成了一个大大的绿色条带。这是因为点击的时候，就添加了一个过滤规则，要求匹配 play\_name 字段里的单词。



你要问了“在哪里过滤了”？

答案就藏在过滤(FILTERING)标签上出现的白色小星星里。点击这个标签，你会发现 filtering 面板里已经添加了一个过滤规则。在 filtering 面板里，可以添加，编辑，固定，删除任意过滤规则。很多面板都支持添加过滤规则，包括表格(table)，直方图(histogram)，地图(map)等等。



过滤规则也可以自己点击 + 号手动添加。

## 更多阅读

你现在已经可以处理过滤和请求了，你可能很好奇在 [Kibana schema](#) 里，他们是怎么存在的。如果你还想知道如何通过 URL 参数来添加请求和过滤，欢迎阅读 [Templated and Scripted Dashboards](#)

# 行和面板

Kibana 的仪表板是由行和面板组成的。这些都可以随意的添加，删除和重组。

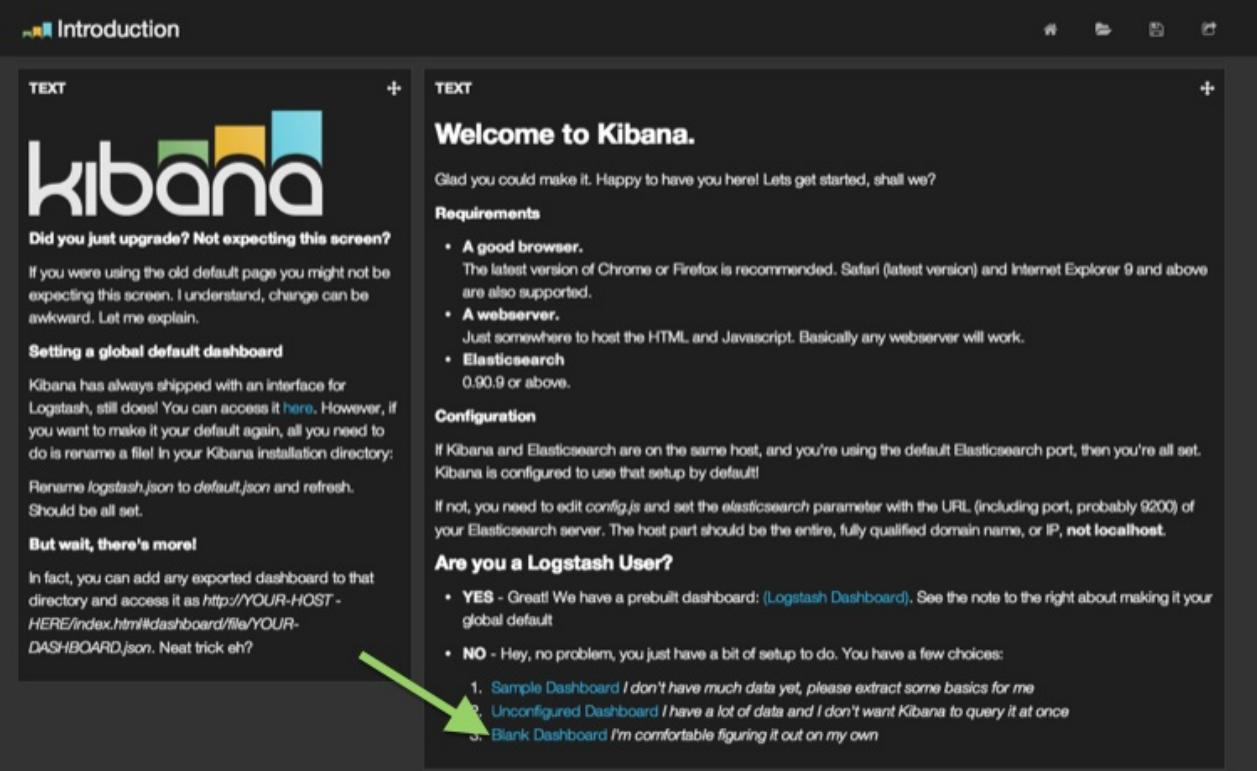
这节我们会介绍：

- 加载一个空白仪表板
- 添加，隐藏行，以及修改行高
- 添加面板和修改面板宽度
- 删除面板和行

我们假设你已经：

- 在自己电脑上安装好了 Elasticsearch
- 在自己电脑上搭建好了网站服务器，并把 Kibana 发行包解压到了发布目录里
- 读过 [Using Kibana for the first time](#) 并且按照文章内容准备好了存有莎士比亚文集的索引

## 加载一个空白仪表板

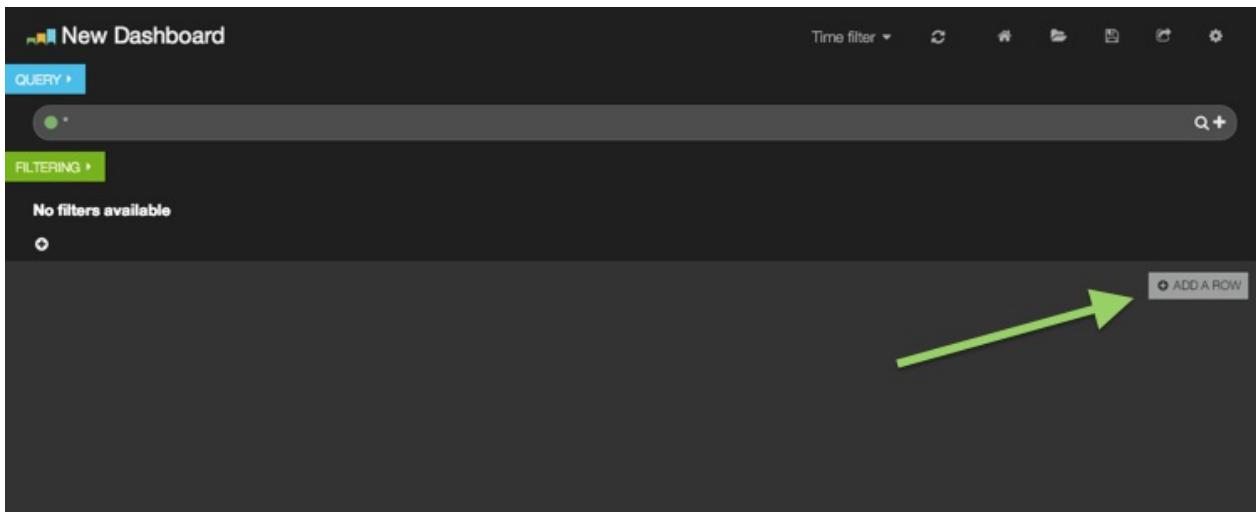


The screenshot shows the Kibana 'Introduction' page. On the left, there's a sidebar with sections like 'TEXT' and 'Requirements'. The main content area has a heading 'Welcome to Kibana.' followed by a paragraph of text. Below that is a 'Requirements' section with a bulleted list. At the bottom right of the main content area, there's a list of options under 'Are you a Logstash User?'. A green arrow points from the bottom of the sidebar towards this list. The options in the list are:

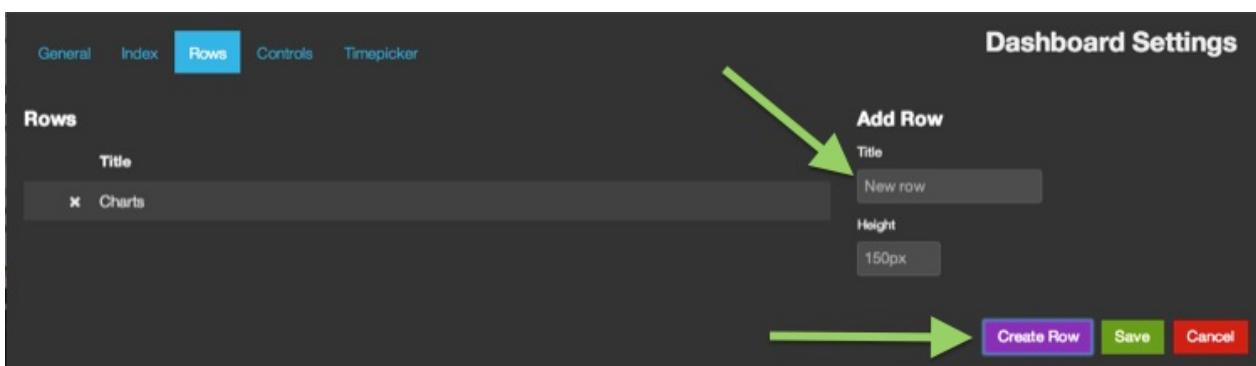
- YES - Great! We have a prebuilt dashboard: ([Logstash Dashboard](#)). See the note to the right about making it your global default.
- NO - Hey, no problem, you just have a bit of setup to do. You have a few choices:
  1. Sample Dashboard *I don't have much data yet, please extract some basics for me*
  2. Unconfigured Dashboard *I have a lot of data and I don't want Kibana to query it at once*
  3. Blank Dashboard *I'm comfortable figuring it out on my own*

从主屏里选择第三项，就会加载一个空白仪表板(Blank Dashboard)。默认情况下，空白仪表板会搜索 Elasticsearch 的 \_all 紴引，也就是你的全部索引。要指定搜索某个索引的，阅读 [Using Kibana for the first time](#)。

## 添加一行



你的新空白仪表板上只有展开的请求和过滤区域，页面顶栏上有个时间过滤选择器，除此以外什么都没有。在右下方，点击添加行(ADD A ROW)按钮，添加你的第一行。

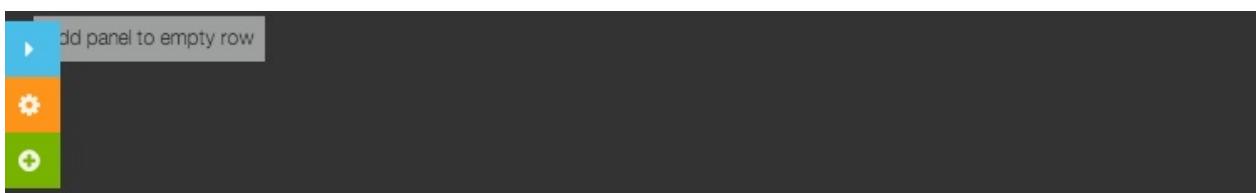


给你的行取个名字，然后点击创建(Create Row)按钮。你会看到你的新行出现在左侧的行列表里。点击保存(Save)

## 行的控制



现在你有了一行，你会注意到仪表板上多了点新元素。主要是左侧多出来的三个小小的不同颜色的长方形。移动鼠标到它们上面



哈哈！看到了吧，这三个按钮是让你做这三件事情的：

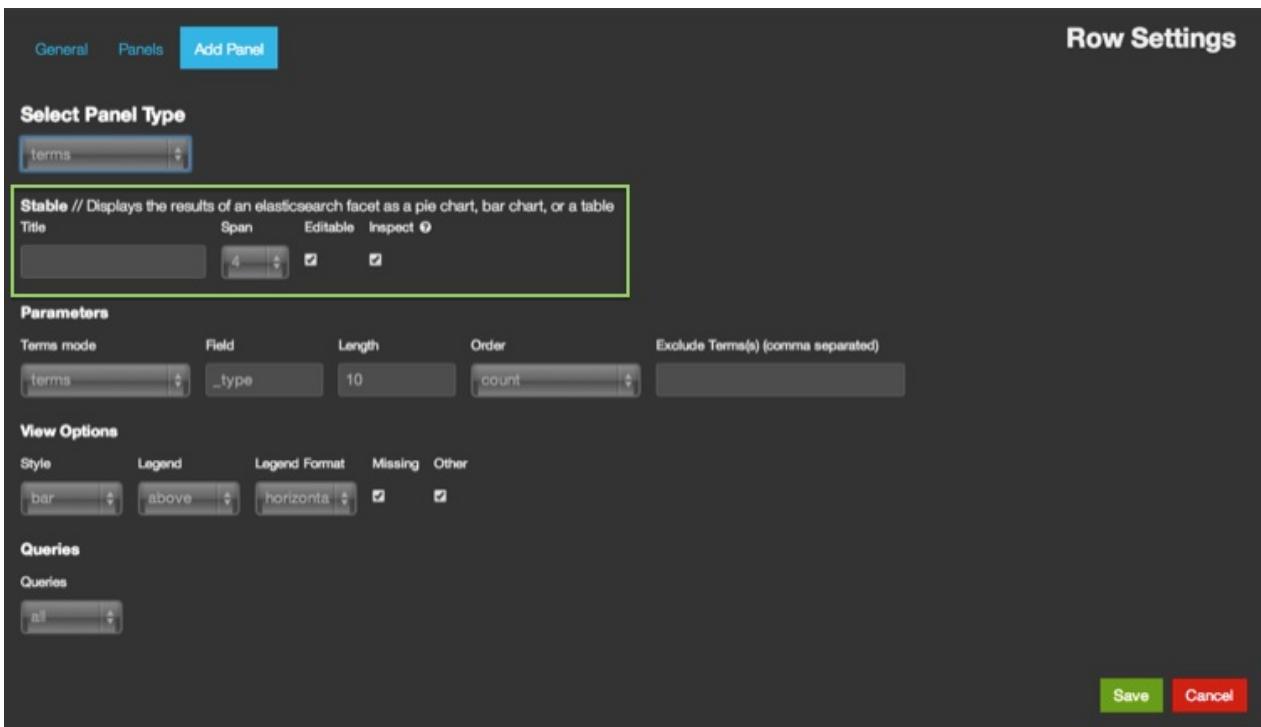
- 折叠行(蓝色)
- 配置行(橘色)
- 添加面板(绿色)

## 添加面板

现在我们专注在行控制力的绿色按钮上，试试点击它。你也可以点击空白行内的灰色按钮(Add panel to empty row)，不过它是灰色的啊，有啥意思……



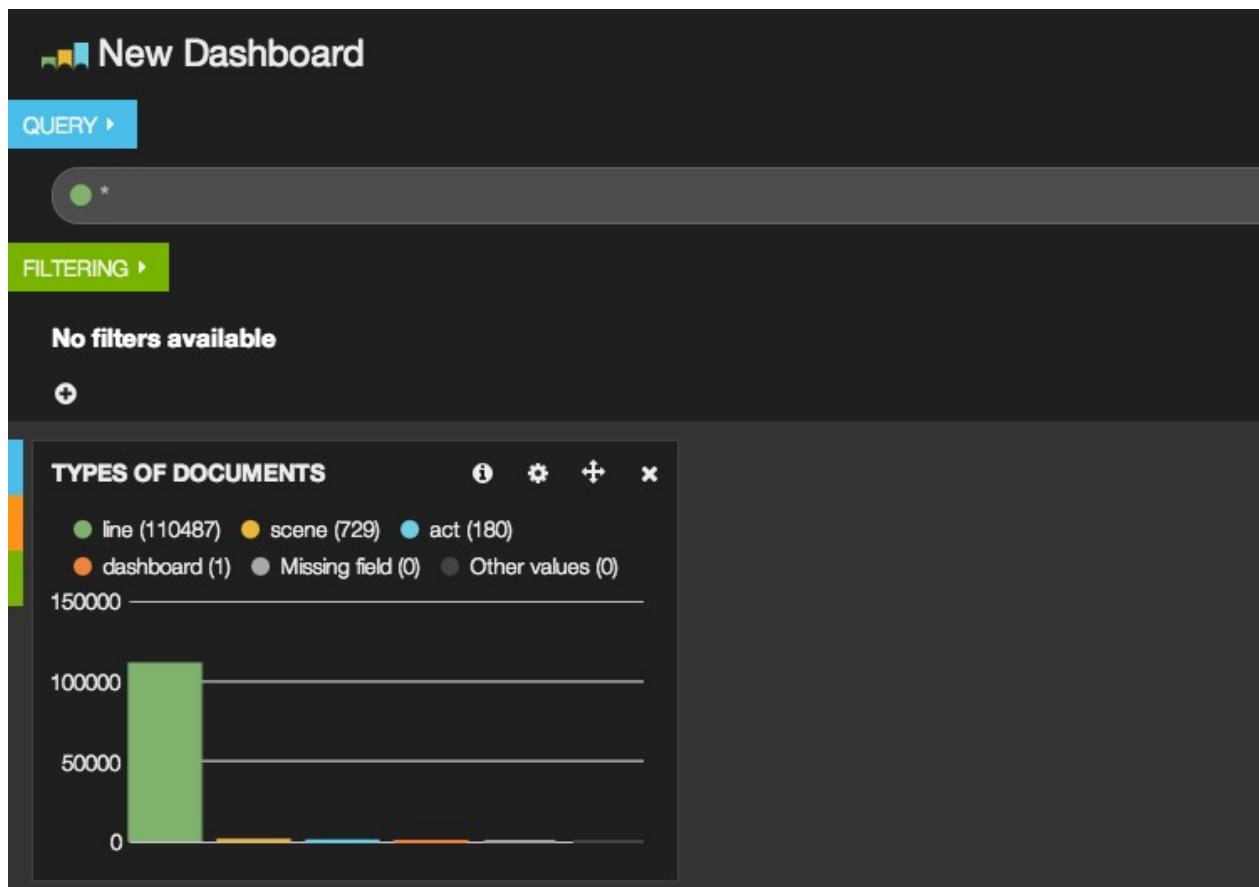
让我们来添加一个 terms 面板。terms 面板可以让我们用上 Elasticsearch 的 terms facet 功能，查找一个字段内最经常出现的几个值。



你可以看到，terms 面板有一系列可配置选项，不过我们现在先只管第一段里德通用配置好了：

1. Title: 面板的名称
2. Span: 面板的宽度。Kibana 仪表板等分成 12 个 spans 面板最大就是到 12 个 spans 宽。但是行可以容纳超过 12 个 spans 的总宽度，因为它会自动把新的面板放到下面显示。现在我们先设置为 4。
3. Editable: 面板是否在之后可以继续被编辑。现在先略过。
4. Inspectable: 面板是否允许用户查看所用的请求内容。现在先略过。
5. 点击 Save 添加你的新 terms 面板到你的仪表板

译者注：面板宽度也可以在仪表板内直接拖拽修改，将鼠标移动至面板左(右)侧边线处，鼠标会变成相应的箭头，按住左键拖拽成满意宽度松开即可

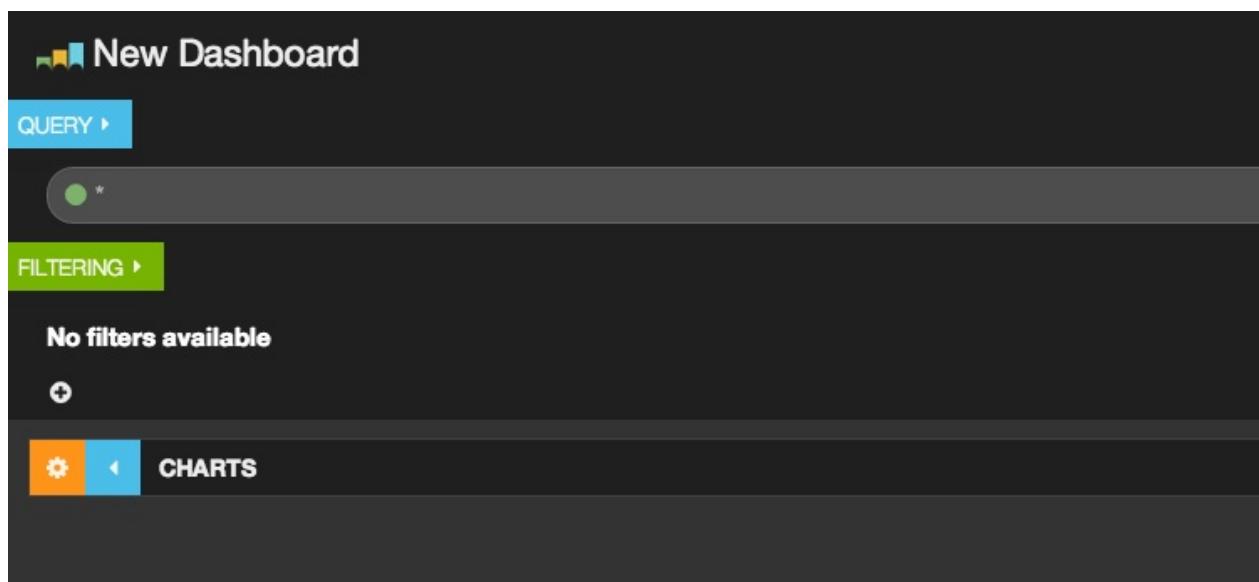


太棒了！你现在有一个面板了！你可能意识到这个数据跟 [Using Kibana for the first time](#) 中的饼图数据一样。shakespeare 数据集集中在 lines，还有少量的 acts 和 scenes。

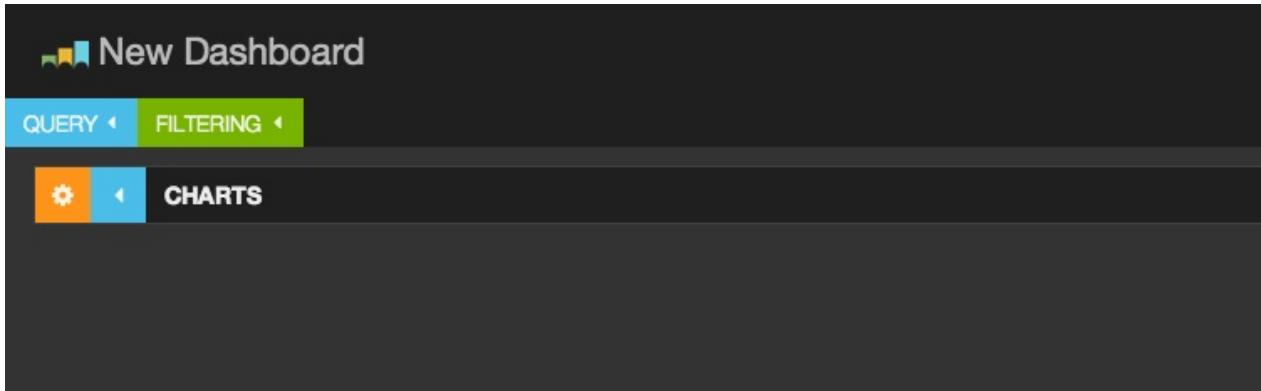
## 折叠和展开行



蓝色按钮可以折叠你的行。被折叠行里的面板不会刷新数据，也就不要求 Elasticsearch 资源。所以折叠行可以用于那些你不需要经常看的数据。有需要的时候点击蓝色按钮展开就可以了。

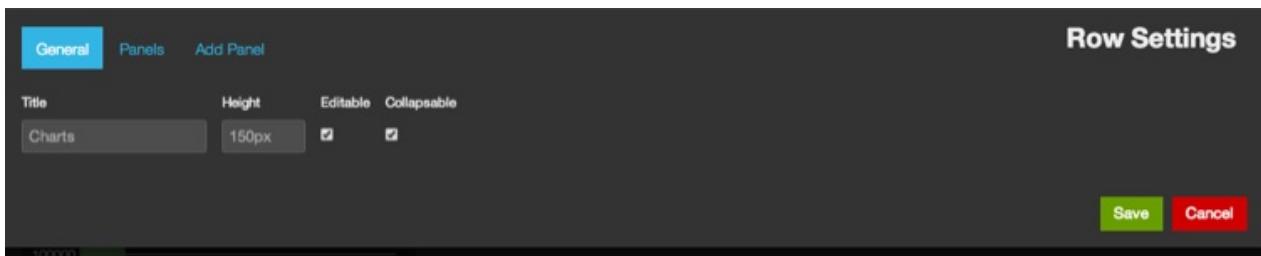


顶部的请求和过滤区域也可以被折叠。点击彩色标签就可以折叠和展开。

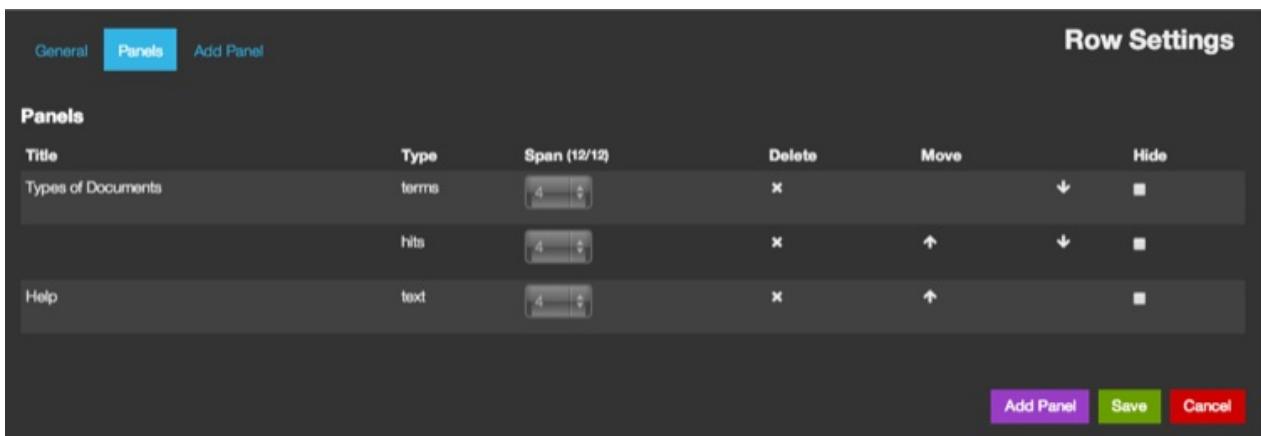


## 编辑行

通过行编辑器，可以给行重命名，改行高等其他配置。点击橙色按钮打开行编辑器。

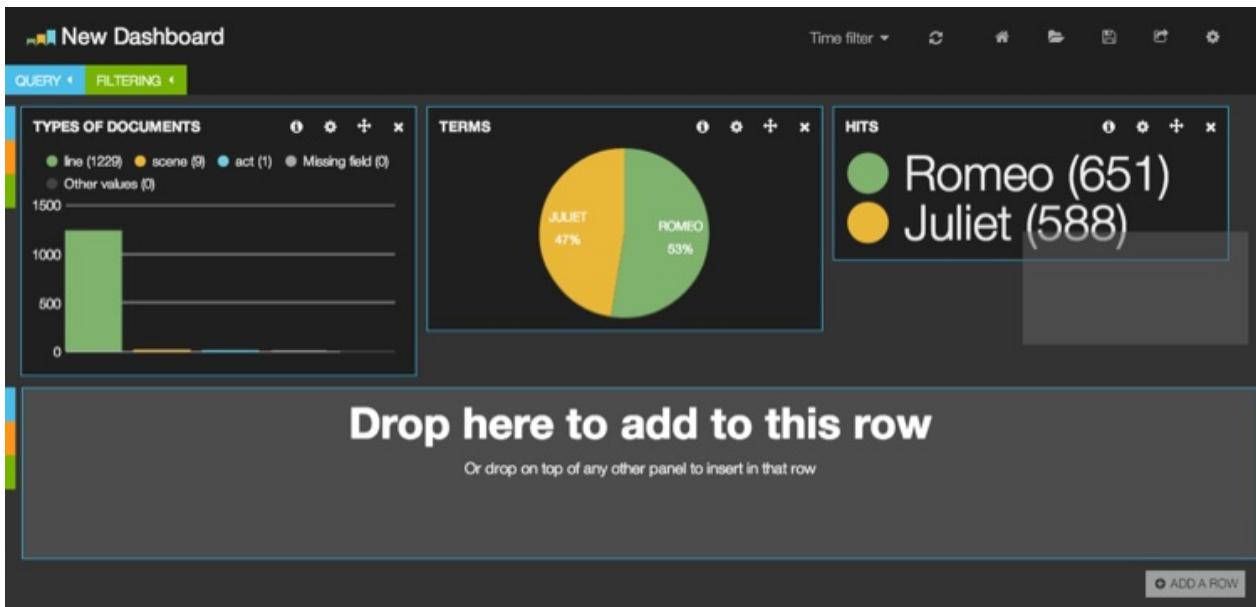


这个对话框还允许你修改面板的排序和大小，以及删除面板。

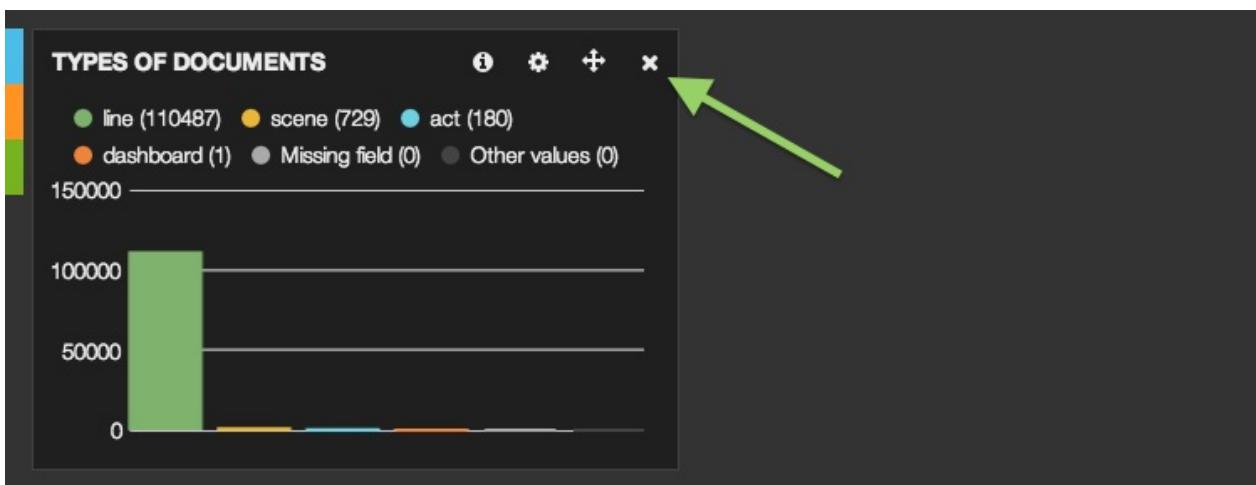


## 移动和删除面板

面板可以在本行，甚至其他行之间任意拖拽。按住面板右上角的十字架形状小图标然后拖动即可。

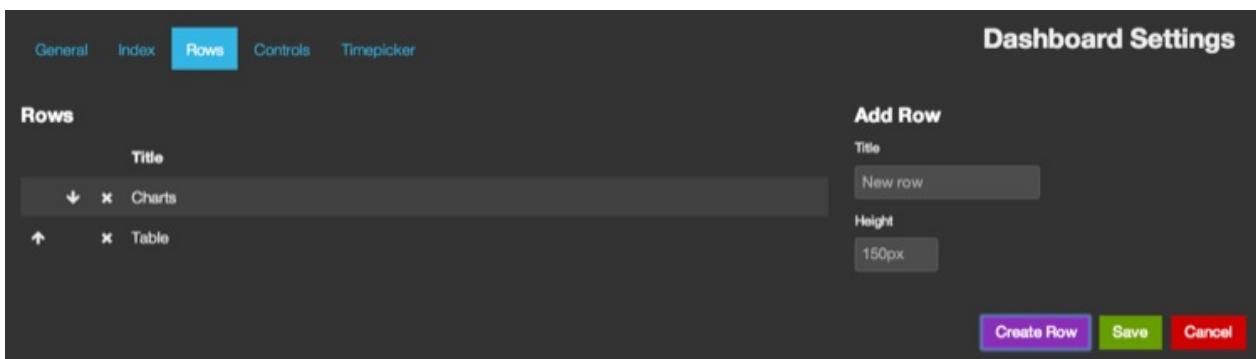


点击面板右上角的 remove 小图标就可以从仪表板上移除它。前面说到从行编辑器上也可以做到统一效果。



## 移动和删除行

行可以在仪表板配置页中重新排序和删。点击屏幕右上角的配置按钮，选择行(Rows)标签切换到行配置层。看到这里你一定会记起来我们在添加第一个行时候的屏幕。



左侧的箭头用来修改仪表板上行的次序。X 用来删除行。

## 下一步

在你关闭浏览器之前，你可能打算保存这个新仪表板。请阅读 [Saving and Loading dashboards](#)。

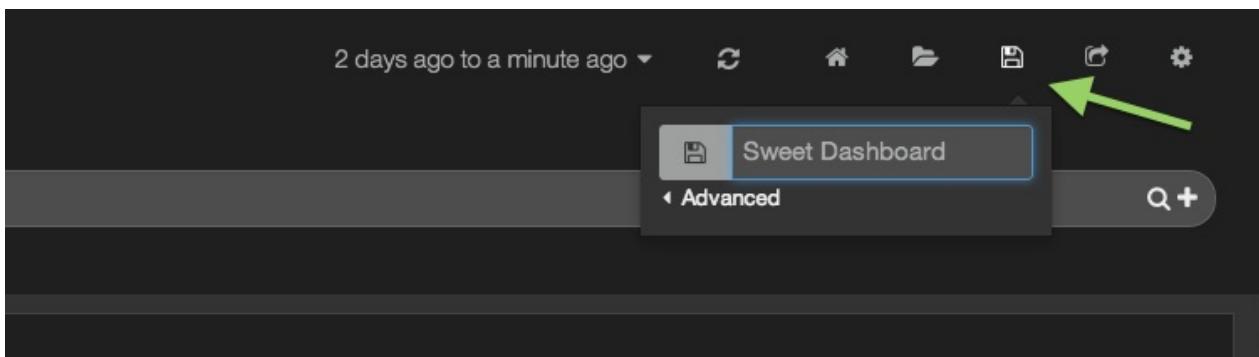
# 保存和加载

你已经构建了一个漂亮的仪表板！现在你打算分享给团队，或者开启自动刷新后挂在一个大屏幕上？Kibana 可以把仪表板设计持久化到 Elasticsearch 里，然后在需要的时候通过加载菜单或者 URL 地址调用出来。



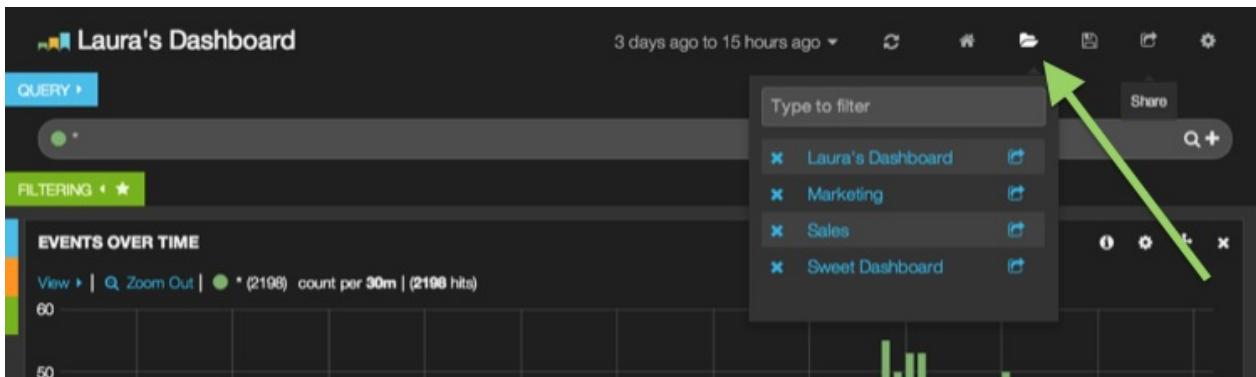
## 保存你漂亮的仪表板

保存你的界面非常简单，打开保存下拉菜单，取个名字，然后点击保存图表即可。现在你的仪表板就保存在一个叫做 `kibana-int` 的 Elasticsearch 索引里了。



## 调用你的仪表板

要搜索已保存的仪表板列表，点击右上角的加载图标。在这里你可以加载、分享和删除仪表板。



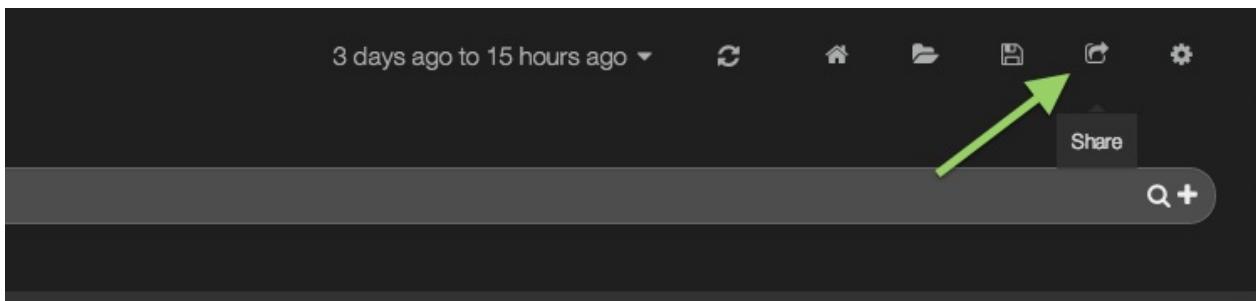
## 分享仪表板

已保存的仪表板可以通过你浏览器地址栏里的 URL 分享出去。每个持久化到 Elasticsearch 里的仪表板都有一个对应的 URL，像下面这样：

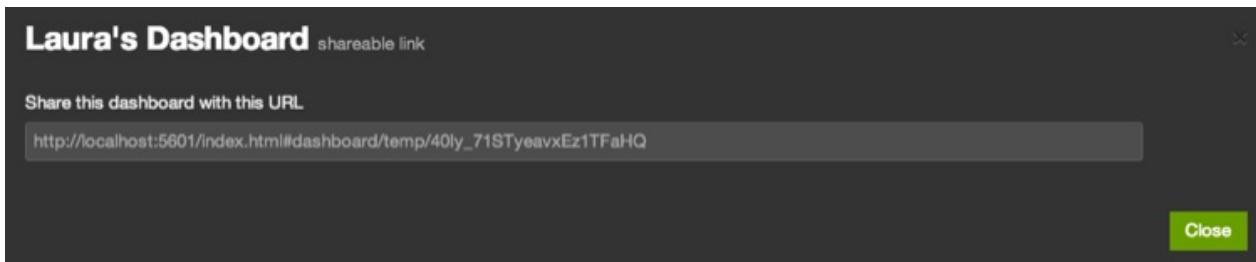
```
http://your_host/index.html#/dashboard/elasticsearch/MYDASHBOARD
```

这个示例中 `MYDASHBOARD` 就是你在保存的时候给仪表板取得名字。

你还可以分享一个即时的仪表板链接，点击 Kibana 右上角的分享图标，会生成一个临时 URL。



默认情况下，临时 URL 保存 30 天。



## 保存成静态仪表板

仪表板可以保存到你的服务器磁盘上成为 `.json` 文件。把文件放到 `app/dashboards` 目录，然后通过下面地址访问

```
http://your_host/index.html#/dashboard/file/MYDASHBOARD.json
```

`MYDASHBOARD.json` 就是磁盘上文件的名字。注意路径中得 `/#/dashboard/file/` 看起来跟之前访问保存在 Elasticsearch 里的仪表板很类似，不过这里访问的是文件而不是 Elasticsearch。导出的仪表板纲要的详细信息，阅读 [The Dashboard Schema Explained](#)

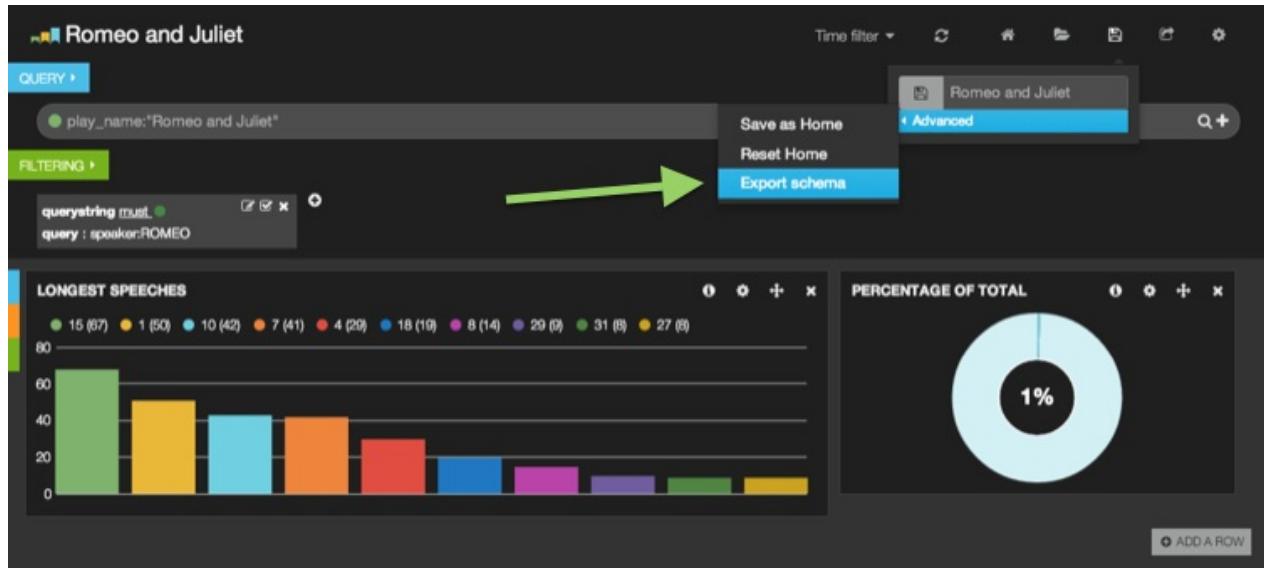
## 下一步

你现在知道怎么保存，加载和访问仪表板了。你可能想知道怎么通过 URL 传递参数来访问，这样可以在其他应用中直接链接过来。请阅读 [Templated and Scripted Dashboards](#)

# 仪表板纲要

Kibana 仪表板可以很容易的在浏览器中创建出来，而且绝大多数情况下，浏览器已经足够支持你创建一个很有用很丰富的节目了。不过，当你真的需要一点小修改的时候，Kibana 也可以让你直接编辑仪表板的纲要。

注意：本节内容只针对高级用户。JSON 语法非常严格，多一个逗号，少一个大括号，都会导致你的仪表板无法加载。



我们会用上面这个仪表板作为示例。你可以导出任意的仪表板纲要，点击右上角的保存按钮，指向高级(Advanced)菜单，然后点击导出纲要(Export Schema)。示例使用的纲要文件可以在这里下载：[schema.json](#)

因为仪表板是由特别长的 JSON 文档组成的，我们只能分成一段段的内容，分别介绍每段的作用和目的。

和所有的 JSON 文档一样，都是以一个大括号开始的。

```
{
```

## 服务(services)

```
"services": {
```

服务(Services)是被多个面板使用的持久化对象。目前仪表板对象附加有 2 种服务对象，不指明的话，就会自动填充成请求(query)和过滤(filter)服务了。

- Query
- Filter

### query

```
"query": {  
  "list": {  
    "0": {  
      "query": "play_name:\\"Romeo and Juliet\\\"",  
      "alias": "",  
      "color": "#7EB26D",  
      "id": 0,  
      "pin": false,  
      "type": "lucene",  
      "enable": true
```

```
        },
    },
    "ids": [
        0
    ]
},
```

请求服务主要是由仪表板顶部的请求栏控制的。有两个属性：

- List: 一个以数字为键的对象。每个值描述一个请求对象。请求对象的键命名一目了然，就是描述请求输入框的外观和行为的。
- Ids: 一个由 ID 组成的数组。每个 ID 都对应前面 list 对象的键。ids 数组用来保证显示时 list 的排序问题。

## filter

```
"filter": {
    "list": {
        "0": {
            "type": "querystring",
            "query": "speaker:ROMEO",
            "mandate": "must",
            "active": true,
            "alias": "",
            "id": 0
        }
    },
    "ids": [
        0
    ]
},
```

过滤的行为和请求很像，不过过滤不能在面板级别选择，而是对全仪表板生效。过滤对象和请求对象一样有 list 和 ids 两个属性，各属性的行为和请求对象也一样。

## 垂幕(pulldown)

```
"pulldowns": [
```

垂幕是一种特殊的面板。或者说，是一个特殊的可以用来放面板的地方。在垂幕里的面板就跟在行里的一样，区别就是不能设置 span 宽度。垂幕里的面板永远都是全屏宽度。此外，垂幕里的面板也不可以吧被使用者移动或编辑。所以垂幕特别适合放置输入框。垂幕的属性是一个由面板对象构成的数组。关于特定的面板，请阅读 [Kibana Panels](#)

```
{
    "type": "query",
    "collapse": false,
    "notice": false,
    "enable": true,
    "query": "*",
    "pinned": true,
    "history": [
        "play_name:\\"Romeo and Juliet\\\"",
        "playname:\\"Romeo and Juliet\\\"",
        "romeo"
    ],
    "remember": 10
},
{
    "type": "filtering",
    "collapse": false,
    "notice": true,
    "enable": true
},
```

垂幕面板有 2 个普通行面板没有的选项：

- Collapse: 设置为真假值，代表着面板被折叠还是展开。
- Notice: 面板设置这个值，控制在垂幕的标签主题上出现一个小星星。用来通知使用者，这个面板里发生变动了。

## 导航(nav)

nav 属性里也有一个面板列表，只是这些面板是被用来填充在页首导航栏里德。目前唯一支持导航的面板是时间选择器 (timepicker)

```
"nav": [
  {
    "type": "timepicker",
    "collapse": false,
    "notice": false,
    "enable": true,
    "status": "Stable",
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ],
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "timefield": "@timestamp"
  }
],
```

## loader

loader 属性描述了仪表板顶部的保存和加载按钮的行为。

```
"loader": {
  "save_gist": false,
  "save_elasticsearch": true,
  "save_local": true,
  "save_default": true,
  "save_temp": true,
  "save_temp_ttl_enable": true,
  "save_temp_ttl": "30d",
  "load_gist": false,
  "load_elasticsearch": true,
  "load_elasticsearch_size": 20,
  "load_local": false,
  "hide": false
},
```

## 行数组

rows 就是通常放置面板的地方。也是唯一可以通过浏览器页面添加的位置。

```
"rows": [
  {
    "title": "Charts",
    "height": "150px",
    "editable": true,
    "collapse": false,
    "collapsable": true,
```

行对象包含了一个面板列表，以及一些行的具体参数，如下所示：

- title: 行的标题
- height: 行的高度，单位是像素，记作 px
- editable: 真假值代表面板是否可被编辑
- collapse: 真假值代表行是否被折叠
- collapsable: 真值代表使用者是否可以折叠行

## 面板数组

行的 panels 数组属性包括有一个以自己出现次序排序的面板对象的列表。各特定面板本身的属性列表和说明，阅读 [Kibana Panels](#)

```
"panels": [
  {
    "error": false,
    "span": 8,
    "editable": true,
    "type": "terms",
    "loadingEditor": false,
    "field": "speech_number",
    "exclude": [],
    "missing": false,
    "other": false,
    "size": 10,
    "order": "count",
    "style": {
      "font-size": "10pt"
    },
    "donut": false,
    "tilt": false,
    "labels": true,
    "arrangement": "horizontal",
    "chart": "bar",
    "counter_pos": "above",
    "spyable": true,
    "queries": {
      "mode": "all",
      "ids": [
        0
      ]
    },
    "tmode": "terms",
    "tstat": "total",
    "valuefield": "",
    "title": "Longest Speeches"
  },
  {
    "error": false,
    "span": 4,
    "editable": true,
    "type": "goal",
    "loadingEditor": false,
    "donut": true,
    "tilt": false,
    "legend": "none",
    "labels": true,
    "spyable": true,
    "query": {
      "goal": 111397
    },
    "queries": {
      "mode": "all",
      "ids": [
        0
      ]
    }
  }
]
```

```
        ],
      },
      "title": "Percentage of Total"
    }
  ],
},
```

## 索引设置

索引属性包括了 Kibana 交互的 Elasticsearch 索引的信息。

```
"index": {
  "interval": "none",
  "default": "_all",
  "pattern": "[logstash-]YYYY.MM.DD",
  "warm_fields": false
},
```

- `interval`: none, hour, day, week, month。这个属性描述了索引所遵循的时间间隔模式。
- `default`: 如果 `interval` 被设置为 `none`, 或者后面的 `failover` 设置为 `true` 而且没有索引能匹配上正则模式的话, 搜索这里设置的索引。
- `pattern`: 如果 `interval` 被设置成除了 `none` 以外的其他值, 就需要解析这里设置的模式, 启用时间过滤规则, 来确定请求哪些索引。
- `warm_fields`: 是否需要解析映射表来确定字段列表。

## 其余

下面四个也是顶层的仪表板配置项

```
"failover": false,
"editable": true,
"style": "dark",
"refresh": false
}
```

- `failover`: 真假值, 确定在没有匹配上索引模板的时候是否使用 `index.default`。
- `editable`: 真假值, 确定是否在仪表板上显示配置按钮。
- `style`: "亮色(light)" 或者 "暗色(dark)"
- `refresh`: 可以设置为 "false" 或者其他 Elasticsearch 支持的时间表达式(比如 10s, 1m, 1h), 用来描述多久触发一次面板的数据更新。

## 导入纲要

默认是不能导入纲要的。不过在仪表板配置屏的控制(Controls)标签里可以开启这个功能, 启用 "Local file" 选项即可。然后通过仪表板右上角加载图标的高级设置, 选择导入文件, 就可以导入纲要了。

# 模板和脚本

Kibana 支持通过模板或者更高级的脚本来动态的创建仪表板。你先创建一个基础的仪表板，然后通过参数来改变它，比如通过 URL 插入一个新的请求或者过滤规则。

模板和脚本都必须存储在磁盘上，目前不支持存储在 Elasticsearch 里。同时它们也必须是通过编辑或创建纲要生成的。所以我们强烈建议阅读 [The Kibana Schema Explained](#)

## 仪表板目录

仪表板存储在 Kibana 安装目录里的 `app/dashboards` 子目录里。你会注意到这里面有两种文件：`.json` 文件和 `.js` 文件。

### 模板化仪表板(`.json`)

`.json` 文件就是模板化的仪表板。模板示例可以在 `logstash.json` 仪表板的请求和过滤对象里找到。模板使用 handlebars 语法，可以让你在 json 里插入 javascript 语句。URL 参数存在 `ARGS` 对象中。下面是 `logstash.json` ([on github](#)) 里请求和过滤服务的代码片段：

```
"0": {  
  "query": "{{ARGS.query || '*'}}",  
  "alias": "",  
  "color": "#7EB26D",  
  "id": 0,  
  "pin": false  
}  
  
[...]  
  
"0": {  
  "type": "time",  
  "field": "@timestamp",  
  "from": "now-{{ARGS.from || '24h'}}",  
  "to": "now",  
  "mandate": "must",  
  "active": true,  
  "alias": "",  
  "id": 0  
}
```

这允许我们在 URL 里设置两个参数，`query` 和 `from`。如果没设置，默认值就是 `||` 后面的内容。比如说，下面的 URL 就会搜索过去 7 天内 `status:200` 的数据：

注意：千万注意 url `#/dashboard/file/logstash.json` 里的 `file` 字样

```
http://yourserver/index.html#/dashboard/file/logstash.json?query=status:200&from=7d
```

### 脚本化仪表板(`.js`)

脚本化仪表板比模板化仪表板更加强大。当然，功能强大随之而来的就是构建起来也更复杂。脚本化仪表板的目的就是构建并返回一个描述了完整的仪表板纲要的 javascript 对象。`app/dashboards/logstash.js` ([on github](#)) 就是一个有着详细注释的脚本化仪表板示例。这个文件的最终结果和 `logstash.json` 一致，但提供了更强大的功能，比如我们可以以逗号分割多个请求：

注意：千万注意 URL `#/dashboard/script/logstash.js` 里的 `script` 字样。这让 kibana 解析对应的文件为 javascript 脚本。

```
http://yourserver/index.html#/dashboard/script/logstash.js?query=status:403,status:404&from=7d
```

这会创建 2 个请求对象，`status:403` 和 `status:404` 并分别绘图。事实上这个仪表板还能接收另一个参数 `split`，用于指定用什么字符串切分。

```
http://yourserver/index.html#/dashboard/script/logstash.js?query=status:403!status:404&from=7d&split=!
```

我们可以看到 `logstash.js` ([on github](#)) 里是这么做的：

```
// In this dashboard we let users pass queries as comma separated list to the query parameter.
// Or they can specify a split character using the split parameter
// If query is defined, split it into a list of query objects
// NOTE: ids must be integers, hence the parseInt()
if(!_.isUndefined(ARGs.query)) {
  queries = _.object(_.map(ARGs.query.split(ARGs.split||','), function(v,k) {
    return [k,{
      query: v,
      id: parseInt(k,10),
      alias: v
    }];
  }));
} else {
  // No queries passed? Initialize a single query to match everything
  queries = {
    0: {
      query: '*',
      id: 0,
    }
  };
}
```

该仪表板可用参数比上面讲述的还要多，全部参数都在 `logstash.js` ([on github](#)) 文件里开始的注释中有讲解。

# 配置

---

config.js 是 Kibana 核心配置的地方。文件里包括得参数都是必须在初次运行 kibana 之前提前设置好的。

## 参数

---

### **elasticsearch**

你 elasticsearch 服务器的 URL 访问地址。你应该不会像写个 `http://localhost:9200` 在这，哪怕你的 Kibana 和 Elasticsearch 是在同一台服务器上。默认的时候这里会尝试访问你部署 kibana 的服务器上的 ES 服务，你可能需要设置为你 elasticsearch 服务器的主机名。

注意：如果你要传递参数给 http 客户端，这里也可以设置成对象形式，如下：

```
+elasticsearch: {server: "http://localhost:9200", withCredentials: true}+
```

### **default\_route**

没有指明加载哪个仪表板的时候，默认加载页路径的设置参数。你可以设置为文件，脚本或者保存的仪表板。比如，你有一个保存成 "WebLogs" 的仪表板在 elasticsearch 里，那么你就可以设置成：

```
default_route: /dashboard/elasticsearch/WebLogs,
```

### **kibana-int**

默认用来保存 Kibana 相关对象，比如仪表板，的 Elasticsearch 索引名称。

### **panel\_name**

可用的面板模块数组。面板只有在仪表板中有定义时才会被加载，这个数组只是用在 "add panel" 界面里做下拉菜单。

# 面板

---

**Kibana** 仪表板由面板( `panels` )块组成。面板在行中可以起到很多作用，不过大多数是用来给一个或者多个请求的数据集结果做可视化。剩下一些面板则用来展示数据集或者用来为使用者提供插入指令的地方。

面板可以很容易的通过 Kibana 网页界面配置。为了了解更高级的用法，比如模板化或者脚本化仪表板，这章开始介绍面板属性。你可以发现一些通过网页界面看不到的设置。

每个面板类型都有自己的属性，不过有这么几个是大家共有的。

## **span**

---

一个从 1-12 的数字，描述面板宽度。

## **editable**

---

是否在面板上显示编辑按钮。

## **type**

---

本对象包含的面板类型。每个具体的面板类型又要求添加新的属性，具体列表说明稍后详述。

---

## 译作者注

---

官方文档中只介绍了各面板的参数，而且针对的是要使用模板化，脚本化仪表板的高级用户，其中有些参数甚至在网页界面上根本不可见。

为了方便初学者，我将在每个面板的参数介绍之后，自行添加界面操作和效果方面的说明。

# histogram

---

状态：稳定

histogram 面板用以显示时间序列图。它包括好几种模式和变种，用以显示时间的计数，平均数，最大值，最小值，以及数值字段的和，计数器字段的导数。

## 参数

---

### 轴(axis)参数

- mode

用于 Y 轴的值。除了 count 以外，其他 mode 设置都要求定义 value\_field 参数。可选值为：count, mean, max, min, total。我的 fork 中新增了一个可选值为 uniq。

- time\_field

X 轴字段。必须是在 Elasticsearch 中定义为时间类型的字段。

- value\_field

如果 mode 设置为 mean, max, min 或者 total，Y 轴字段。必须是数值型。

- x-axis

是否显示 X 轴。

- y-axis

是否显示 Y 轴。

- scale

以该因子规划 Y 轴

- y\_format

Y 轴数值格式，可选：none, bytes, short

### 注释

- 注释对象

可以指定一个请求的结果作为标记显示在图上。比如说，标记某时刻部署代码了。

- annotate.enable

是否显示注释(即标记)

- annotate.query

标记使用的 Lucene query\_string 语法请求

- annotate.size

最多显示多少标记

- `annotate.field`

显示哪个字段

- `annotate.sort`

数组排序，格式为 [field,order]。比如 `[@timestamp,'desc']`，这是一个内部参数。

- `auto_int`

是否自动调整间隔

- `resolution`

如果 `auto_int` 设为真，shoot for this many bars.

- `interval`

如果 `auto_int` 设为假，用这个值做间隔

- `intervals`

在 `view` 选择器里可见的间隔数组。比如 `['auto','1s','5m','3h']`，这是绘图参数。

- `lines`

显示折线图

- `fill`

折线图的区域填充因子，从 1 到 10。

- `linewidth`

折线的宽度，单位为像素

- `points`

在图上显示数据点

- `pointradius`

数据点的大小，单位为像素

- `bars`

显示条带图

- `stack`

堆叠多个序列

- `spyable`

显示审核图标

- `zoomlinks`

显示‘Zoom Out’链接

- `options`

显示快捷的 view 选项区域

- legend

显示图例

- show\_query

如果没设别名(alias), 是否显示请求

- interactive

允许点击拖拽进行放大

- legend\_counts

在图例上显示计数

- timezone

是否调整成浏览器时区。可选值为 : browser, utc

- percentage

把 Y 轴数据显示成百分比样式。仅对多个请求时有效。

- zerofill

提高折线图准确度，稍微消耗一点性能。

- derivative

在 X 轴上显示该点数据在前一个点数据上变动的数值。

- 提示框(tooltip)对象

- tooltip.value\_type

控制 tooltip 在堆叠图上怎么显示，可选值：独立(individual)还是累计(cumulative)

- tooltip.query\_as\_alias

如果没设别名(alias), 是否显示请求

- 网格(grid)对象

Y 轴的最大值和最小值

- grid.min

Y 轴的最小值

- grid.max

Y 轴的最大值

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- o queries.mode

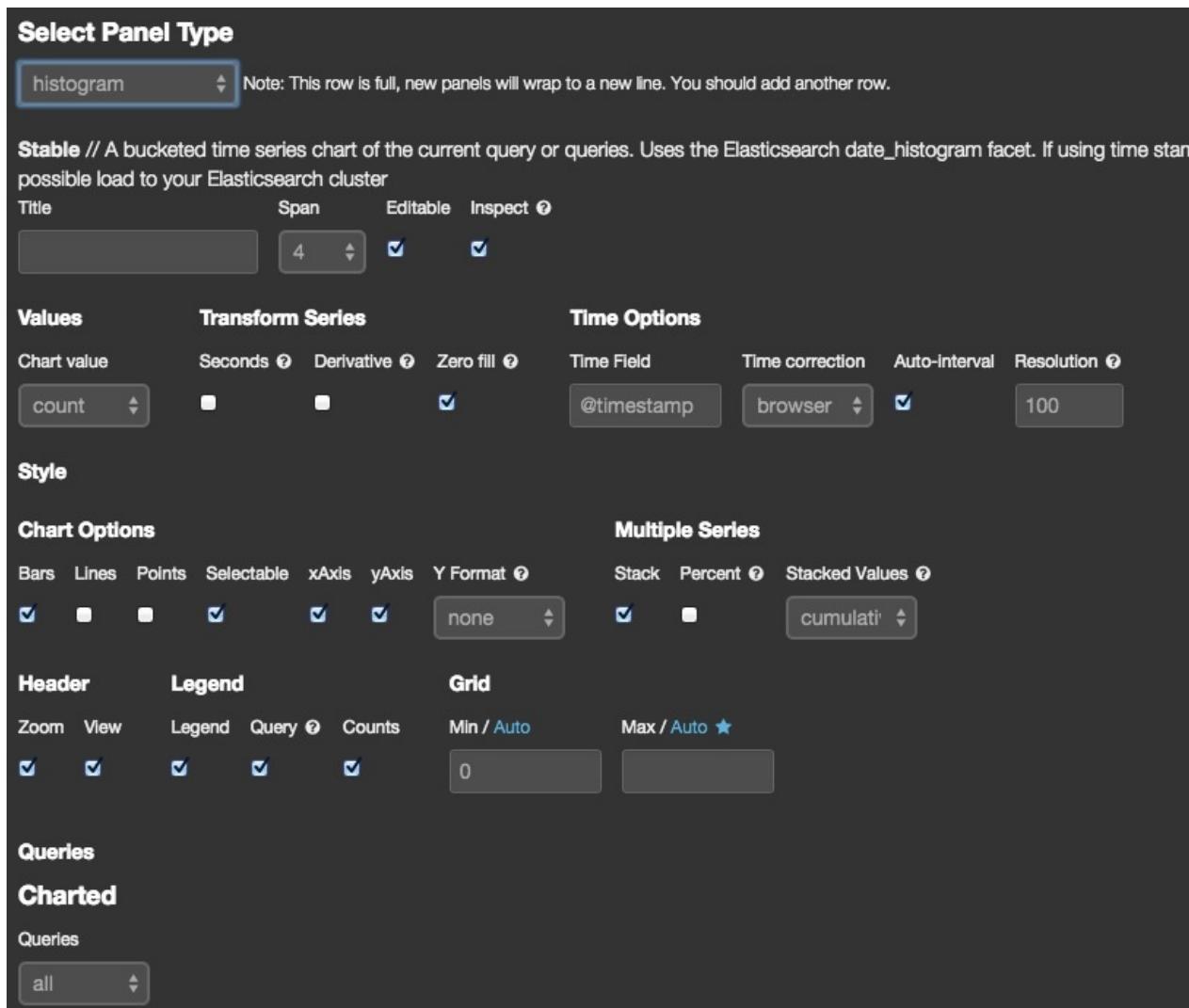
在可用请求中应该用哪些？可设选项有： all, pinned, unpinned, selected

- o queries.ids

如果没为 selected 模式，具体被选的请求编号。

## 界面配置说明

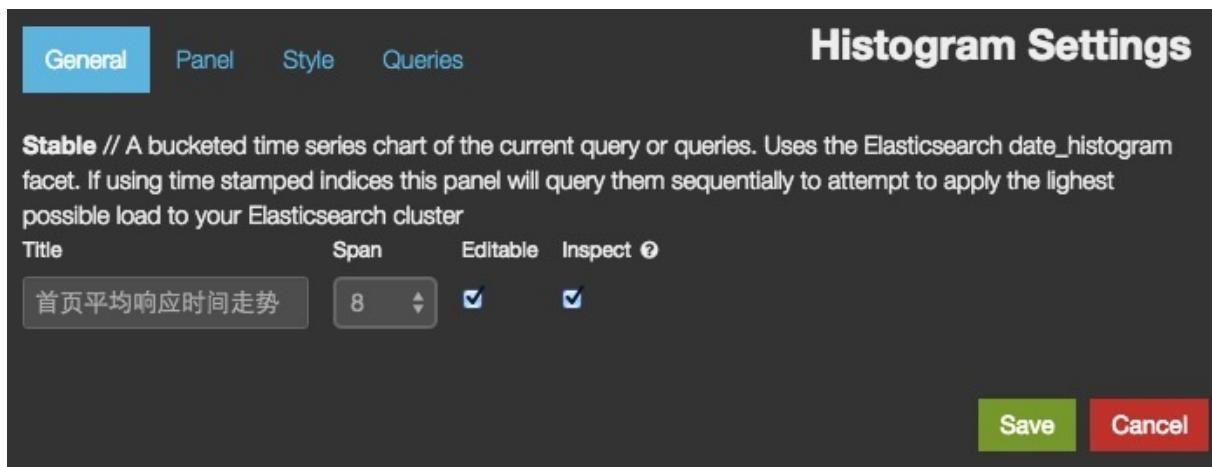
添加面板的方式在之前的“[行和面板](#)”一节中已经有过讲解。在 “Add panel”对话框选择类型为 “histogram” 后，你会看到一系列可配置的选项：



选项分为四类，可以在添加之后，通过点击面板右上角的配置“Configure”小图标弹出浮层继续修改。

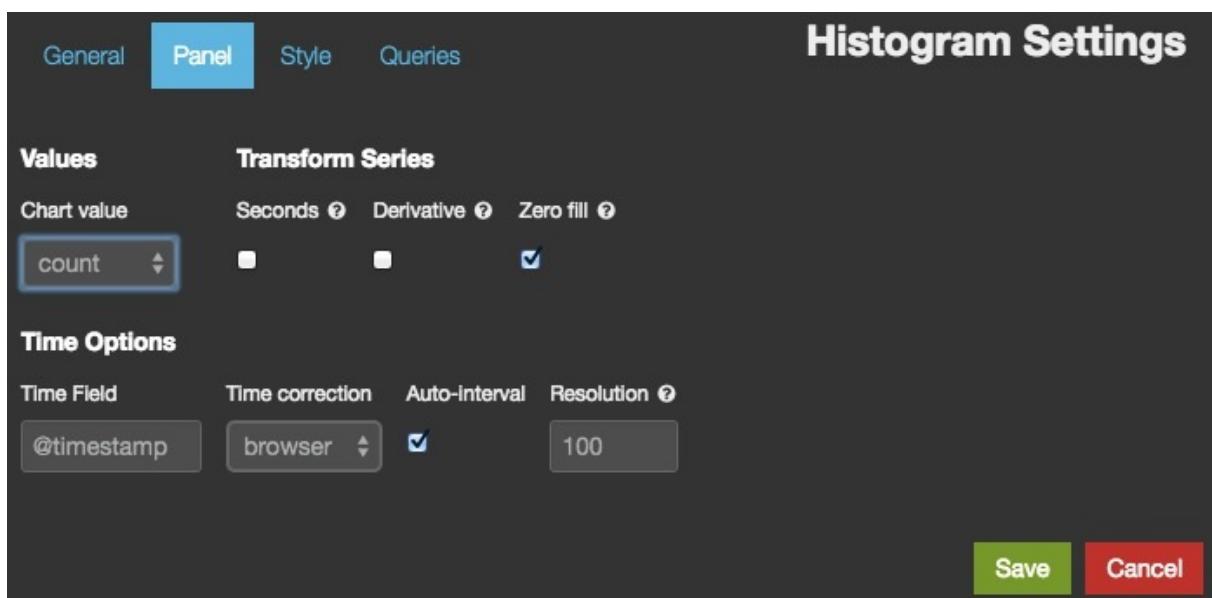
- 通用(General)配置

主要用来修改面板的标题和宽度



- 面板(Panel)配置

设置面板向 Elasticsearch 发出何种请求，以及请求中需要使用的变量。



在 histogram 面板中，经常用的 chart value (即参数部分描述的 mode ) 有：

- count

最常见场景就是统计请求数。这种时候只需要提供一个在 Elasticsearch 中是时间类型的字段(即参数部分描述的 time\_field )即可。一般来说，都是 @timestamp，所以不用修改了。这也是默认的 Logstash 仪表板的基础面板的样式

- mean

最常见场景就是统计平均时间。这时候配置浮层会变成下面这个样子：

**Histogram Settings**

General    **Panel**    Style    Queries

**Values**

Chart value    Value Field [?](#)    Scale    Seconds [?](#)    Derivative [?](#)    Zero fill [?](#)

mean    h5\_view\_loadtime    1           

**Time Options**

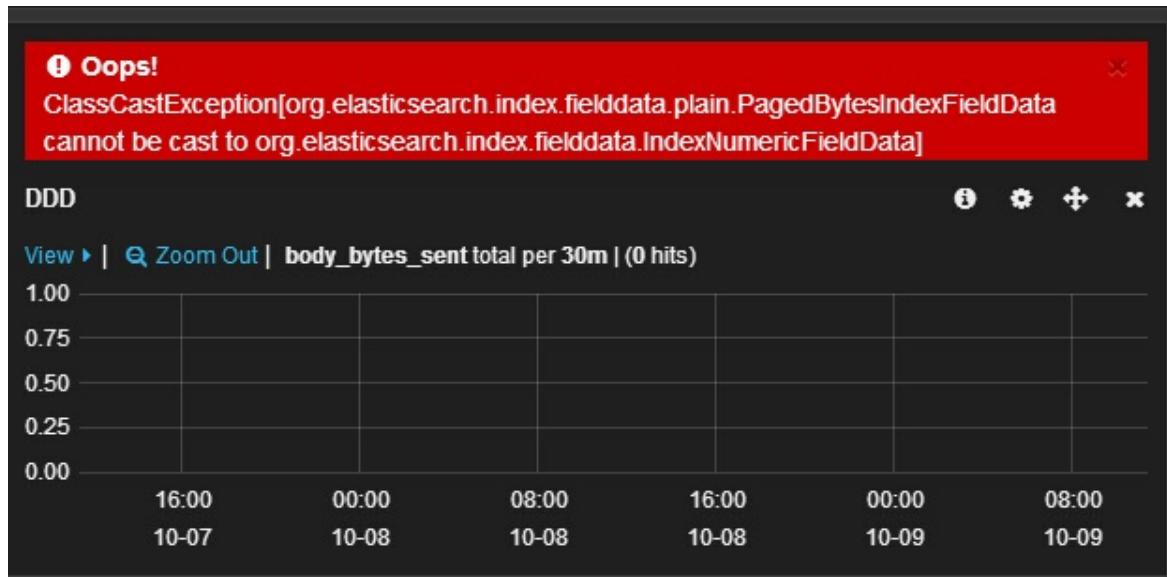
Time Field    Time correction    Auto-Interval    Resolution [?](#)

@timestamp    browser        100

**Save**    **Cancel**

这里就需要提供一个在 Elasticsearch 中是数值类型的字段(即参数部分描述的 `value_field`)作为计算平均值的数据集来源了。以 nginx 访问日志为例，这里就填 "request\_time"。

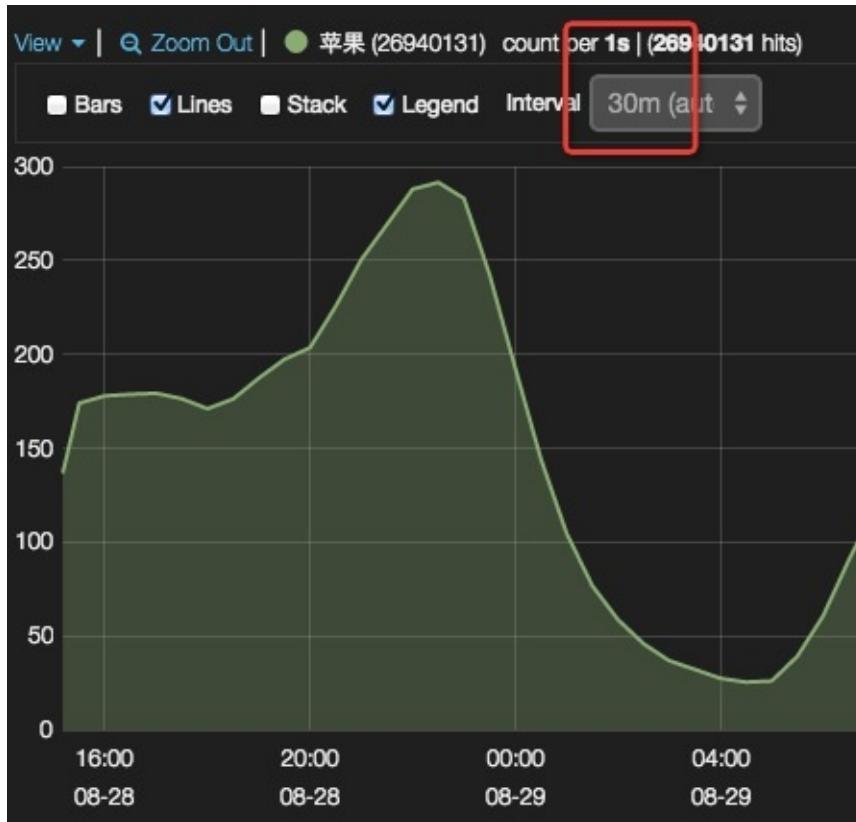
如果你在 Logstash 中使用的是 `%{NUMBER:request_time}`，那么实际类型还是字符串(请记住，正则捕获是 String 类的方法，也只能生成 String 结果)，必须写成 `%{NUMBER:request_time:float}` 强制转换才行。否则，你会看到如下报错信息：



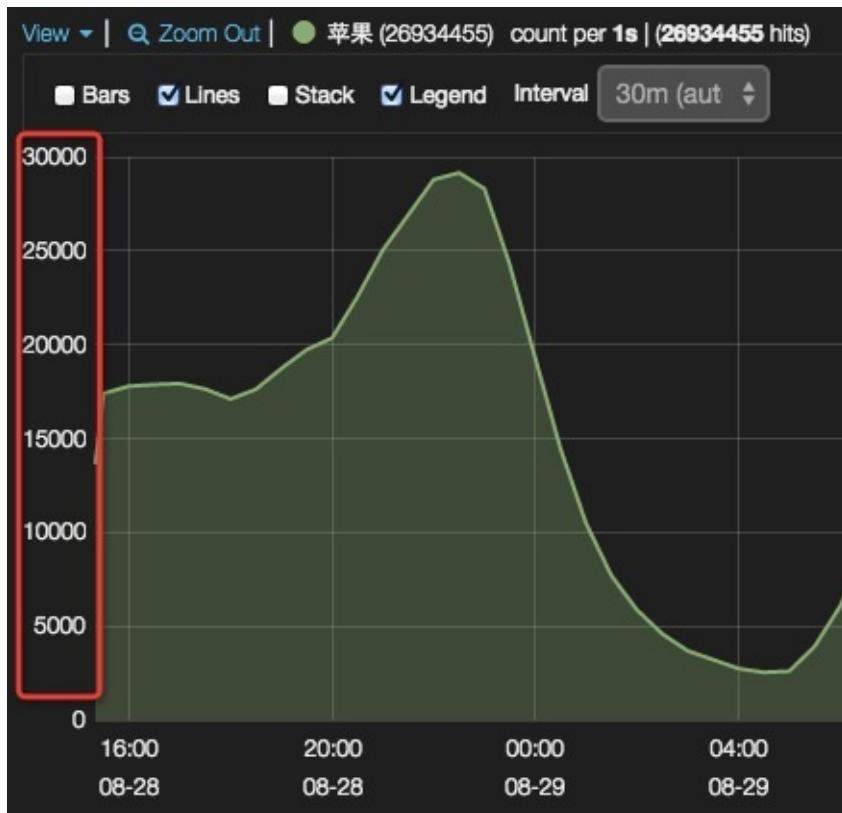
- o total

最常见场景就是统计带宽。配置界面和 mean 是一样的。同样要求填写数值类型的字段名，比如 "bytes\_sent"。

带宽在习惯上会换算成每秒数据，但是通过修改 `interval` 的方式来求每秒数据，对 elasticsearch 性能是一个很大的负担，绘制出来的图形也太过密集影响美观。所以 Kibana 提供了另一种方式：保持 `interval`，勾选 `seconds`。Kibana 会自动将每个数值除以间隔秒数得到每秒数据。( `count` 也可以这样，用来计算 qps 等数据)



另一个有用的功能，假如你的日志量实在太大，被迫采用抽样日志的方式，可以在 Kibana 上填写 `scale`。比如百分之一的抽样日志，`scale` 框就写 100，带宽数据就会在展示的时候自动翻 100 倍显示出来。



- uniq

ES 从 1.1 版本开始通过 HyperLogLog++ 算法支持[去重统计](#)聚合。在用 Aggregation API 重写了 histogram panel 后，也可以支持了。

General    **Panel**    Style    Queries

**Values**

Chart value    Value Field [?](#)

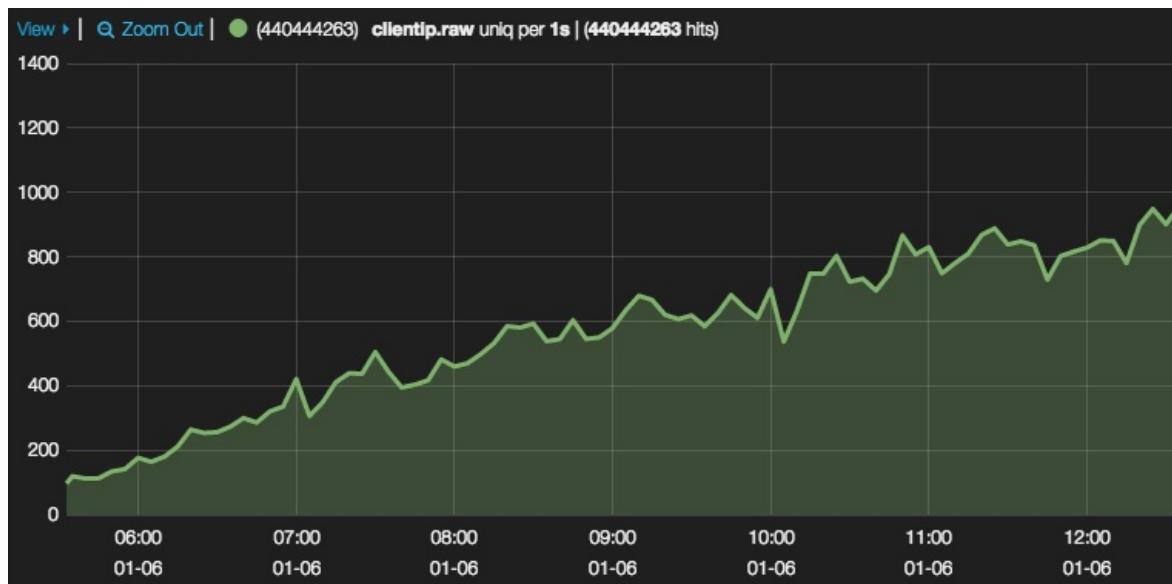
uniq    clientip.raw

**Transform Series**

Scale    Seconds

1   

常用场景比如：UV 统计。效果如下：



- 风格(Style)配置

General    Panel    **Style**    Queries

**Chart Options**

Bars    Lines    Points    Selectable    xAxis    yAxis    Line Fill    Line Width    Y Format [?](#)    Multiple Series

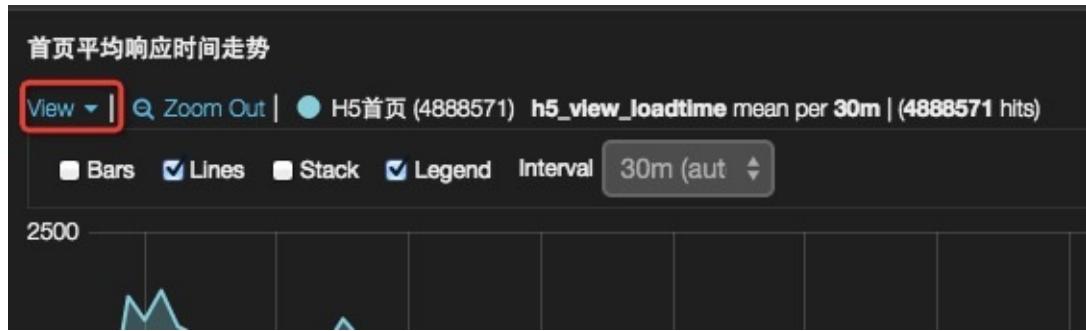
3 2 none

**Header**    **Legend**    **Grid**

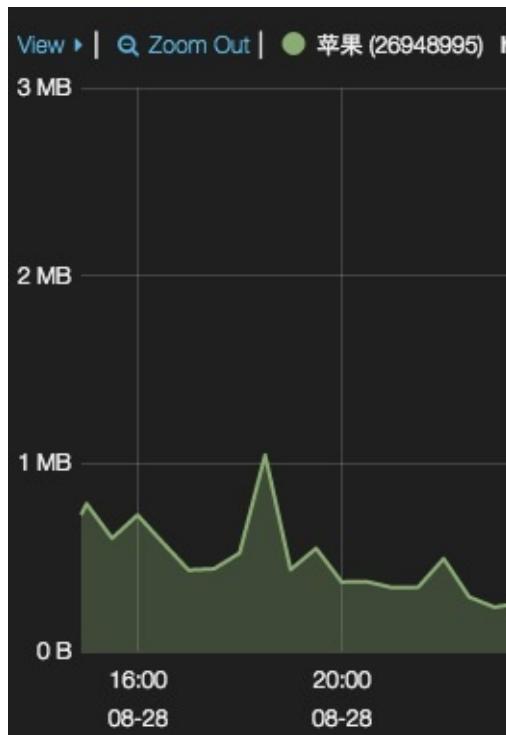
Zoom    View    Legend    Query [?](#)    Counts    Min / Auto    Max / Auto ★

0

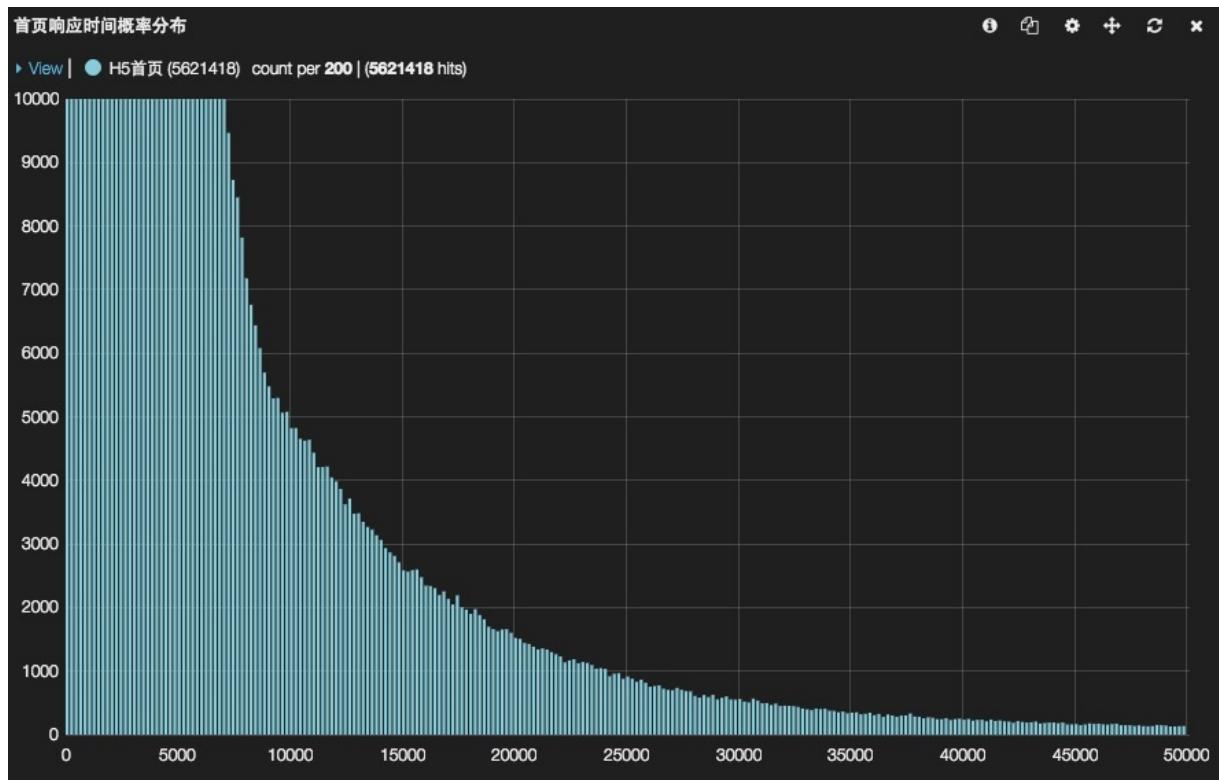
设置获取的数据如何展现。其中小部分(即条带(Bars)、折线(Lines)、散点(Points))可以直接在面板左上角的 "View" 下拉菜单里直接勾选。



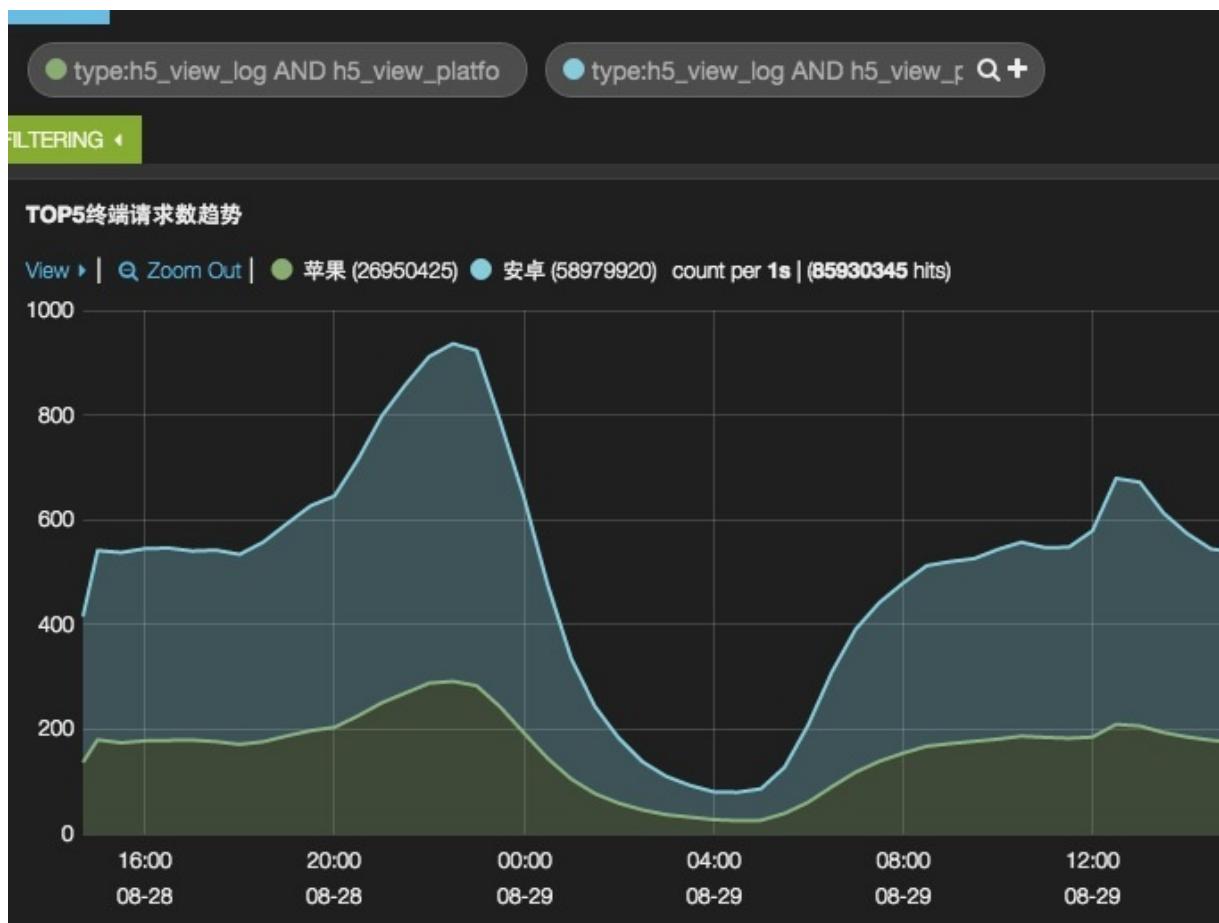
对于带宽数据，可以切换 Y Format 为 bytes。则 Y 轴数据可以自动换算成 MB, GB 的形式，比较方便



此外，还可以在 Grid 区域定义 Y 轴的起始点和终点的具体值。这可以用来在 Y 轴上放大部分区域，观察细微变动；或者忽略某些异常值。



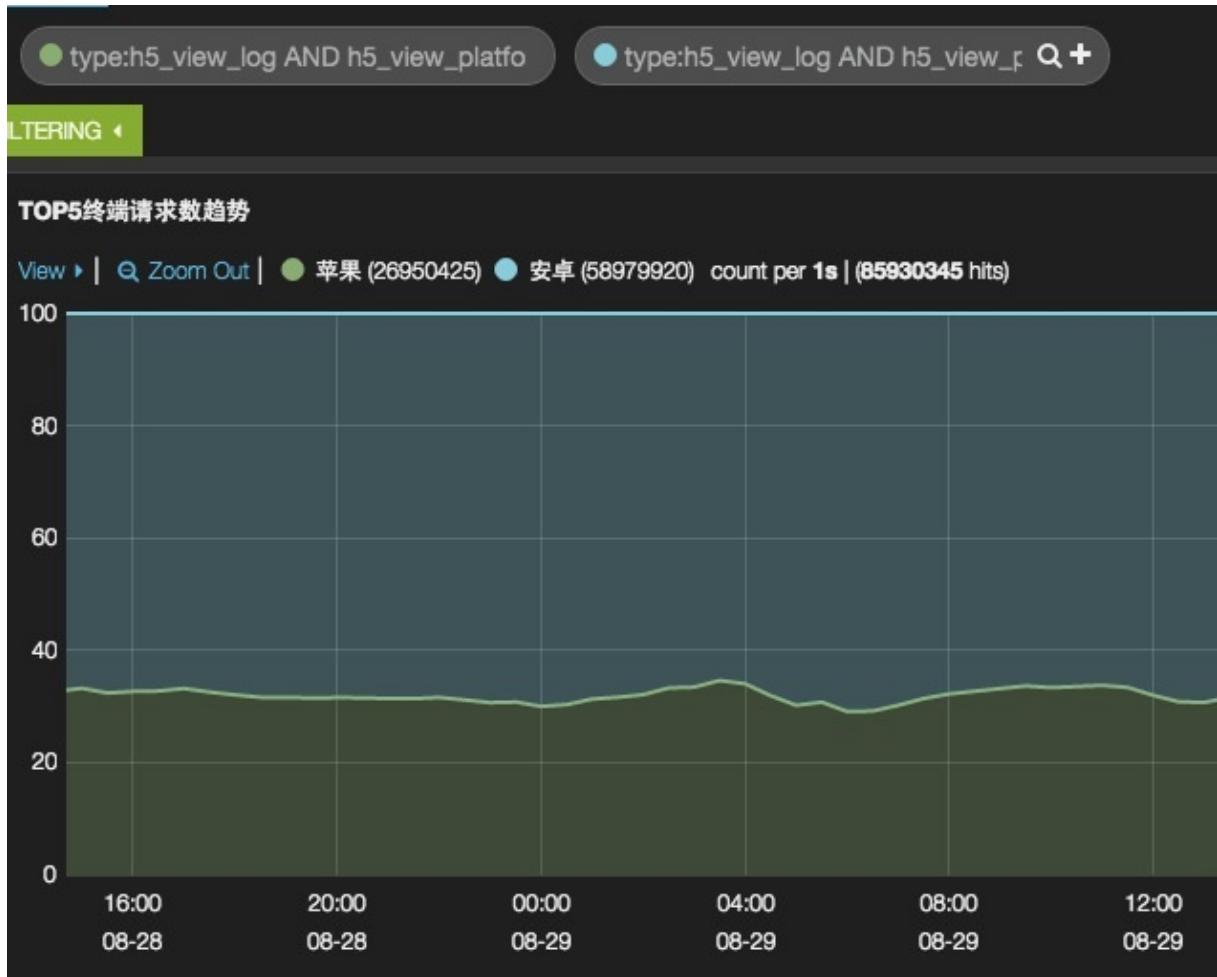
如果面板关联了多个请求，可以勾选以堆叠( Stack )方式展示(最常见的堆叠展示的监控数据就是 CPU 监控)。



堆叠的另一种形式是百分比。在勾选了 Stack 的前提下勾选 Percent 。



效果如下。注意：百分比是  $A / (A + B)$  的值，而不是  $A / B$ 。



- 关联请求(Queries)配置

General Panel Style **Queries**

## Charted

Queries Selected Queries

selected ▾

type:h5\_view\_log  type:h5\_view\_log  总体

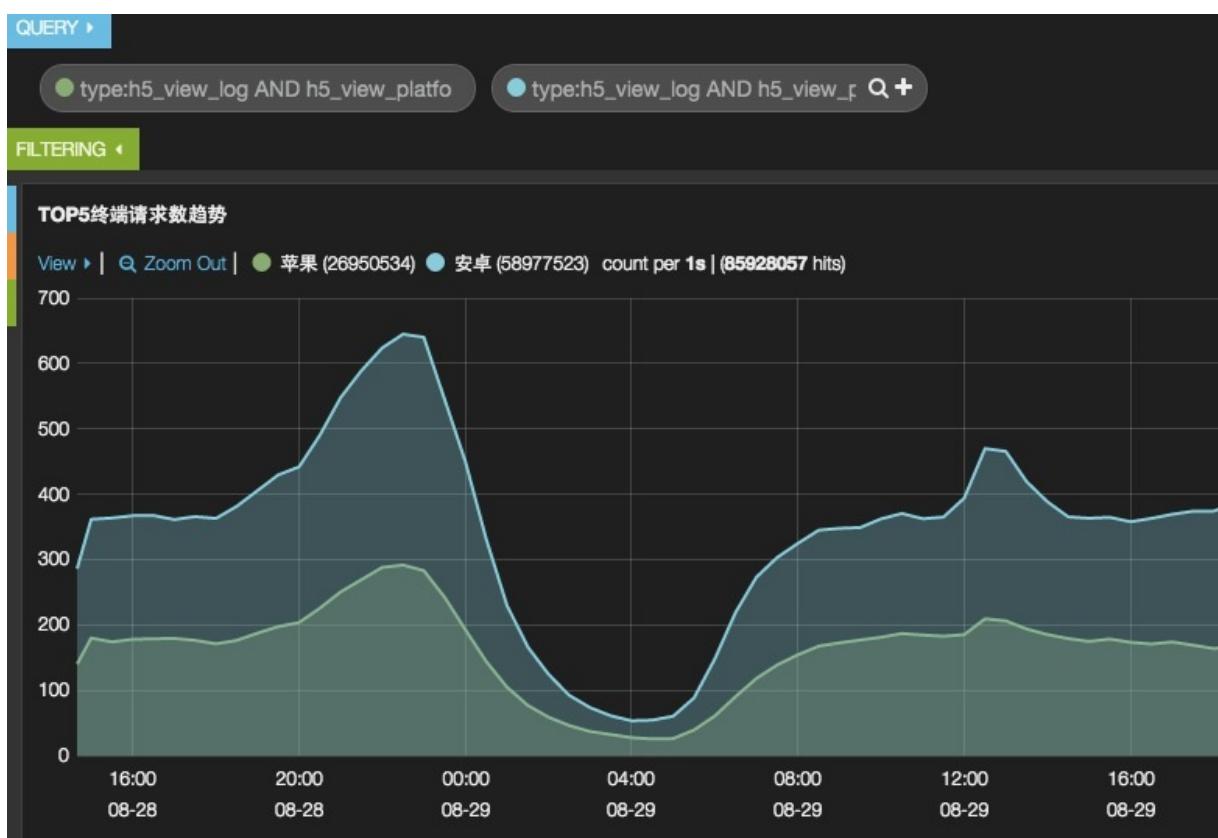
## Markers

Here you can specify a query to be plotted on your chart as a marker. Hovering over a marker will display the field query will be displayed.

Enable

默认的 `Queries` 方式是 `all`。可以使用 `selected` 方式，在右侧选择具体的请求框(可多选)。被选中的会出现边框加粗放大效果。

多请求的默认效果如下。而堆叠和百分比效果，在之前已经谈过。可以对比上下两图的 Y 轴刻度：



# table

---

状态：稳定

表格面板里是一个可排序的分页文档。你可以定义需要排列哪些字段，并且还提供了一些交互功能，比如执行 terms 聚合查询。

## 参数

---

- size

每页显示多少条

- pages

展示多少页

- offset

当前页的页码

- sort

定义表格排序次序的数组，示例如右：['@timestamp', 'desc']

- overflow

css 的 overflow 属性。'min-height' (expand) 或 'auto' (scroll)

- fields

表格显示的字段数组

- highlight

高亮显示的字段数组

- sortable

设为假关掉排序功能

- header

设为假隐藏表格列名

- paging

设为假隐藏表格翻页键

- field\_list

设为假隐藏字段列表。使用者依然可以展开它，不过默认会隐藏起来

- all\_fields

设为真显示映射表内的所有字段，而不是表格当前使用到的字段

- trimFactor

裁剪因子(trim factor)，是参考表格中的列数来决定裁剪字段长度。比如说，设置裁剪因子为 100，表格中有 5 列，那么每列数据就会被裁剪为 20 个字符。完整的数据依然可以在展开这个事件后查看到。

- localTime

设为真调整 `timeField` 的数据遵循浏览器的本地时区。

- timeField

如果 `localTime` 设为真，该字段将会被调整为浏览器本地时区。

- spyable

设为假，不显示审查(inspect)按钮。

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些？可设选项有：all, pinned, unpinned, selected

- queries.ids

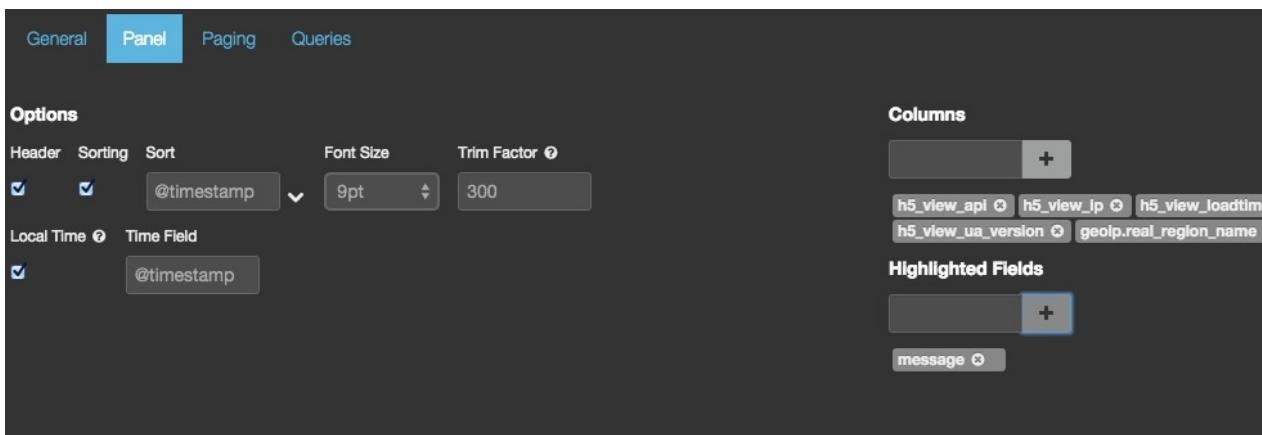
如果设为 `selected` 模式，具体被选的请求编号。

## 界面配置说明

table 和 histogram 面板，是 kibana 默认的 logstash 仪表板里唯二使用的面板。可以说是最重要和常用的组件。

虽然重要，table 面板的可配置项却不多。主要是 panel 和 paging 两部分：

### panel



panel 设置可以分成几类，其中比较重要和有用的是：

- 时间字段

`Time Field` 设置的作用，和 `histogram` 面板中类似，主要是帮助 Kibana 使用者自动转换 elasticsearch 中的 UTC 时间成本

地时间。

- 裁剪因子

和 Splunk 不同，Kibana 在显示事件字段的时候，侧重于单行显示。详情内容通过点击具体某行向下展开的方式参看。每个字段在屏幕中的可用宽度，就会通过裁剪因子来计算。计算方式见官方参数说明部分。

mweibo_cli...	Sep 3 13....	3088603960	com.sina.w...	3.482	/data0/rsy...	102400
View: Table / JSON / Raw						
<b>Field</b> <b>Action</b> <b>Value</b>						
@timestamp	<input type="text"/> ⚡	2014-09-03T05:28:17.000Z				
@version	<input type="text"/> ⚡	1				
_id	<input type="text"/> ⚡	fmv_1Gw9T5un0o-ylj0eYw				
_index	<input type="text"/> ⚡	logstash-2014.09.03				
_type	<input type="text"/> ⚡	mweibo_client_downstream				
agent	<input type="text"/> ⚡	HUAWEI-HUAWEI G610-T00_weibo_4.3.5_android_android4.2.1				
ap	<input type="text"/> ⚡	cmnet				
clientip	<input type="text"/> ⚡	211.140.5.111				
err_msg	<input type="text"/> ⚡	com.sina.weibo.exception.WeiboOException: Invalid response from server: HTTP/1.1 404 Not Found				

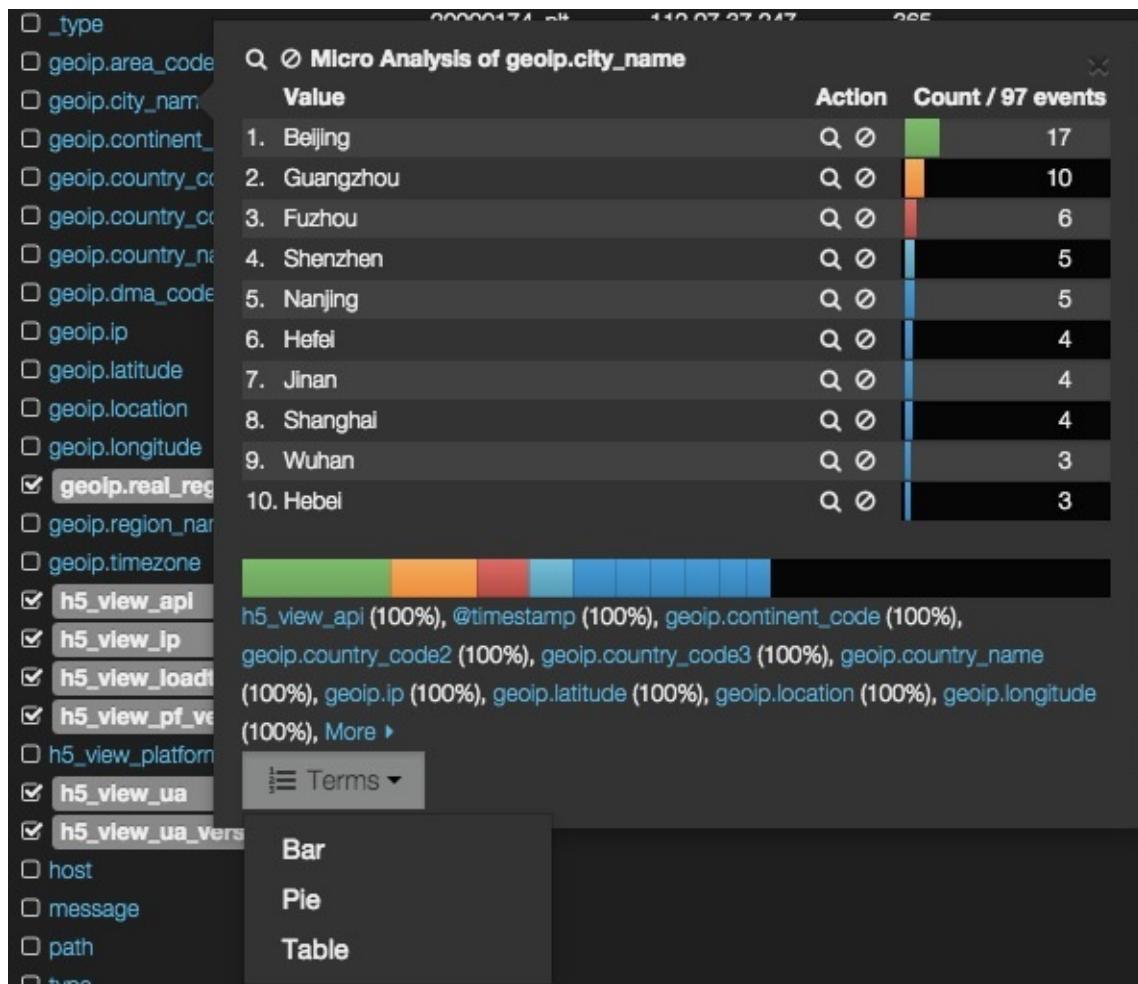
- 字段列表

table 面板左侧，是字段列表多选区域。字段分为 `_all` 和 `current` 两种。`_all` 是 Kibana 通过 elasticsearch 的 `_mapping` API 直接获取的索引内所有存在过的字段；`current` 则仅显示 table 匹配范围内的数据用到的字段。

勾选字段列表中某个字段，该字段就加入 table 面板右侧的表格中成为一列。

Fields	geoip.city_name
<input type="checkbox"/> @timestamp	Beijing
<input type="checkbox"/> @version	Grande Prairie
<input type="checkbox"/> _id	Fuzhou
<input type="checkbox"/> _index	Nanjing
<input type="checkbox"/> _type	Beijing
<input type="checkbox"/> geoip.area_code	Guangzhou
<input checked="" type="checkbox"/> geoip.city_name	Solna
<input type="checkbox"/> geoip.continent_code	Chengdu
<input type="checkbox"/> geoip.country_code2	Shanghai
<input type="checkbox"/> geoip.country_code3	

字段列表中，可以点击具体字段，查看 table 匹配范围内该字段数据的统计和排行数据的小面板。



小面板上虽然只显示一个很小范围内(即size pages, 默认是500)的数据统计, 但是点击小面板底部的 \*TERMS 下拉菜单选项, 生成的 term panel 浮层数据却都是基于整个搜索结果的。这部分的内容介绍。请阅读 [term panel](#) 章节。

- 排序

设置中可以设置一个默认的排序字段。在 logstash 仪表板默认的 event table 中, 设置的是时间字段 @timestamp。不过这个设置, 指的是面板加载的时候, 使用该字段排序, 实际你可以在表头任意字段名上单击, 以该字段的值来临时排序。排序字段会在表头本列字段名后, 出现一个小三角图标, 三角箭头朝上代表升序, 反之降序:

0 to 20 of 100 available			
h5_view_api ▾	h5_view_ip ▾	h5_view_loadtime ▾ ▾	h5_view_pf_version ▾
20000174_plt	123.165.83.31	3	Android4.4.2
20000174_plt	36.43.121.234	10	other
20000174_plt	221.205.158.178	11	other
20000174_plt	117.184.185.46	12	other
20000174_plt	202.106.55.226	13	other
20000174_plt	202.106.55.226	14	other
20000174_plt	121.2.75.176	14	iPhone50
20000174_plt	121.32.24.163	15	other
20000174_plt	202.106.55.226	15	other

- 高亮

elasticsearch 作为一种搜索引擎，很贴心的提供了高亮功能。Kibana 中也同样支持解析 ES 返回的 HTML 高亮文本。只需要在 panel 标签页右侧添加 `Highlighted field`，在搜索框里填入的关键词，如果出现在被指定为 `Highlighted field` 的字段里，这个词在 table 里就会高亮显示(前提是该字段已经在字段列表中勾选)。

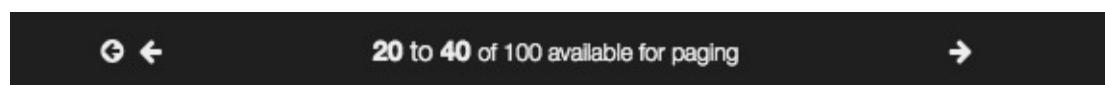
The screenshot shows the Kibana interface with a search query for "Windows". The results table has a header "message" and contains four rows of log entries. The word "Windows" is highlighted in red in each row. The table includes pagination controls at the bottom.

message
web169.mweibo.yf.sinanode.com[_unread]-60031 1409623500589 58.30.134.15 Windows other Chrome Chrome_31.0.1650.48
web163.mweibo.yf.sinanode.com[_unread]-59820 1409498259506 66.222.223.236 Windows other Chrome Chrome_21.0.1180.91
web033.mweibo.bx.sinanode.com[_unread]-59461 1409641384203 218.5.2.198 Windows other Chrome Chrome_21.0.1180.91
web110.mweibo.yf.sinanode.com[10000011_pit]-12085 1409537731330 58.215.136.64 Windows other UC UC_9.9.2.467

小贴士：高亮仅在 table 状态有效，点击展开后的事件详情中是不会高亮的。

## paging

考虑到同时展示太多内容，一来对 elasticsearch 压力较大，二来影响页面展示效果和渲染性能。



# map

---

状态：稳定

map 面板把 2 个字母的国家或地区代码转成地图上的阴影区域。目前可用的地图包括世界地图，美国地图和欧洲地图。

## 参数

---

- map

显示哪个地图 : world, usa, europe

- colors

用来涂抹地图阴影的颜色数组。一旦设定好这 2 个颜色，阴影就会使用介于这 2 者之间的颜色。示例 ['#A0E2E2', '#265656']

- size

阴影区域的最大数量

- exclude

排除的区域数组。示例 ['US', 'BR', 'IN']

- spyable

设为假，不显示审查(inspect)按钮。

### 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些？可设选项有： all, pinned, unpinned, selected

- queries.ids

如果设为 selected 模式，具体被选的请求编号。

---

## 界面配置说明

---

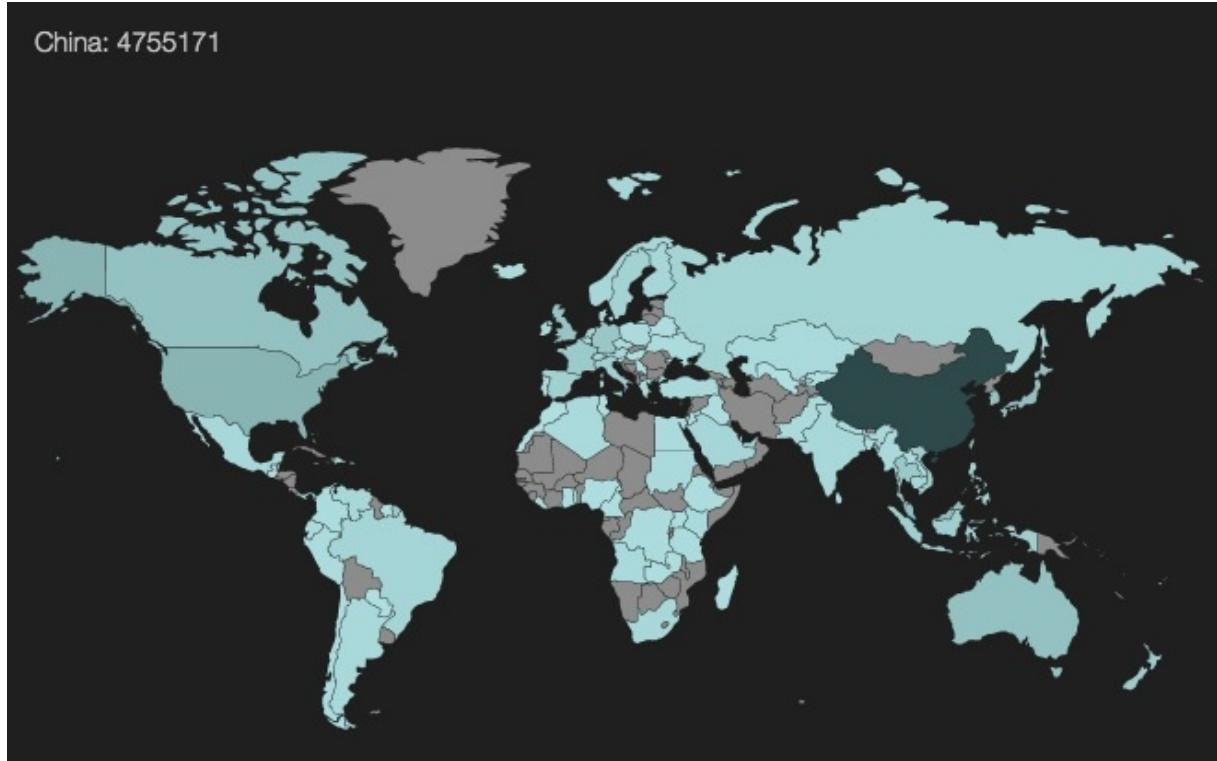
考虑到都是中国读者，本节准备采用中国地图进行讲解，中国地图代码，，基于本人 <https://github.com/chenrynkibana.git> 仓库，除标准的 map 面板参数外，还提供了 terms\_stats 功能，也会一并讲述。

map 面板，最重要的配置即输入字段，对于不同的地图，应该配置不同的 Field :

- world

对于世界地图，其所支持的格式为由 2 个字母构成的国家名称缩写，比如： us , cn , jp 等。如果你使用了 Logstash::Filters::GeoIP 插件，那么默认生成的 geoip.country\_code2 字段正好符合条件。

China: 4755171



- cn

对于中国地图，其所支持的格式则是由 2 个数字构成的省份编码，比如：01 (即安徽)，30 (即广东)，04 (即江苏)等。如果你使用了 `Logstash::Filters::GeoIP` 插件，那么默认生成的 `geoip.region_name` 字段正好符合条件。



如果你使用了我的仓库代码，或者自行合并了[该功能](#)，你的 map 面板配置界面会稍有变动成下面样子

## Select Panel Type

map

Note: This row is full, new panels will wrap to a new line. You should add another row.

**Stable** // Displays a map of shaded regions using a field containing a 2 letter country , or US state, code. Regions with more hit are important that you set it to the correct field.

Title      Span      Editable      Inspect ?

4



### Parameters

terms  
✓ terms\_stats

Field ?

Value field

Stats type of Value

Length ?

Map

100

world

### Queries

Queries

all

如果选择 terms\_stats 模式，就会和 histogram 面板一样出现需要填写 value\_field 的位置。同样必须使用在 Elasticsearch 中是数值类型的字段，然后显示的地图上，就不再是个数而是具体的均值，最大值等数据了。



# bettermap

---

状态：实验性

Bettermap之所以叫 bettermap 是因为还没有更好(better)的名字。Bettermap 使用地理坐标来在地图上创建标记集群，然后根据集群的密度，用橘色，黄色和绿色作为区分。

要查看细节，点击标记集群。地图会放大，原有集群分裂成更小的集群。一直小到没法成为集群的时候，单个标记就会显现出来。悬停在标记上可以查看 `tooltip` 设置的值。

注意: bettermap 需要从互联网上下载它的地图面板文件。

## 参数

---

- `field`

包含了地理坐标的字段，要求是 geojson 格式。GeoJSON 是一个数组，内容为 `[longitude, latitude]`。这可能跟大多数实现(`[latitude, longitude]`)是反过来的。

- `size`

用来绘制地图的数据集大小。默认是 1000。注意：table panel 默认是展示最近 500 条，跟这里的大小不一致，可能引起误解。

- `spyable`

设为假，不显示审查(inspect)按钮。

- `tooltip`

悬停在标记上时显示哪个字段。

- `provider`

选择哪家地图提供商。

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- `queries.mode`

在可用请求中应该用哪些？可设选项有：`all, pinned, unpinned, selected`

- `queries.ids`

如果没为 `selected` 模式，具体被选的请求编号。

---

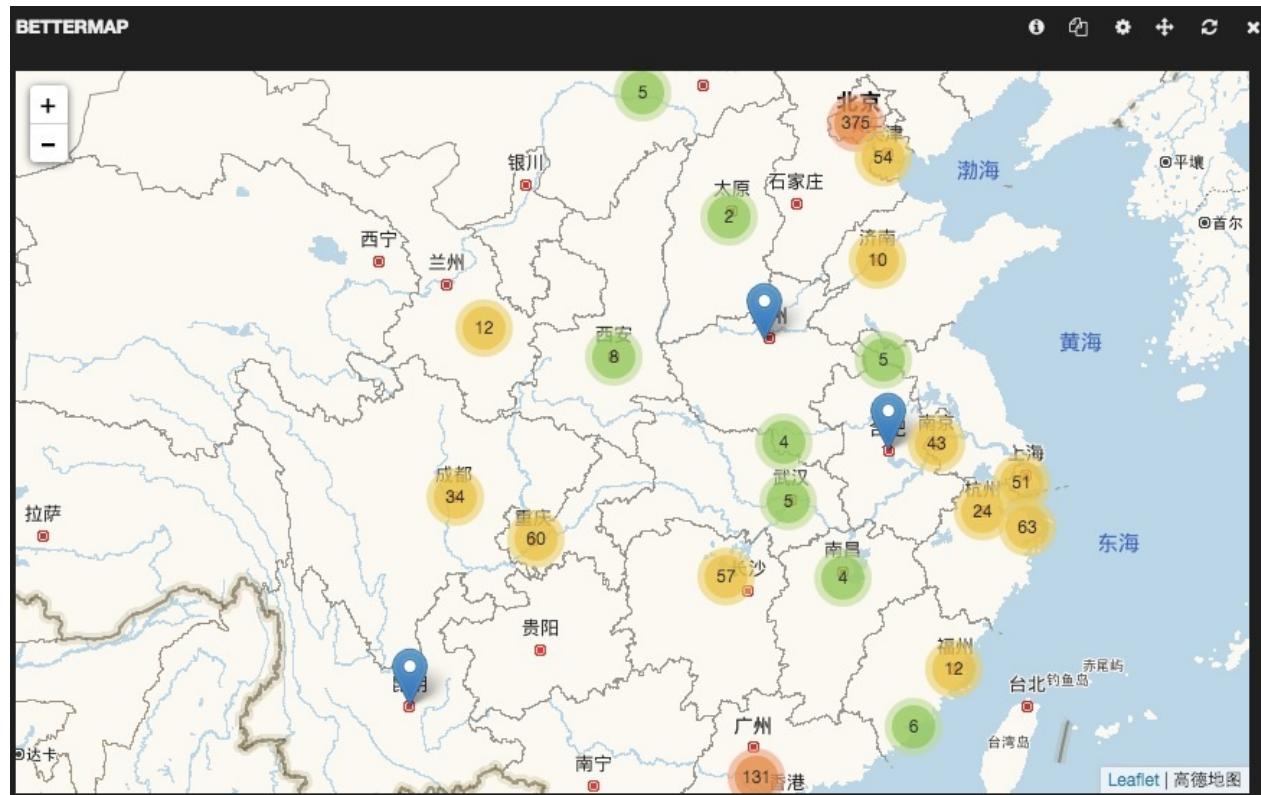
## 界面配置说明

---

bettermap 面板是为了解决 map 面板地图种类太少且不方便大批量添加各国地图文件的问题开发的。它采用了 [leaflet 库](#)，其 `L.tileLayer` 加载的 OpenStreetMap 地图文件都是在使用的时候单独请求下载，所以在初次使用的时候会需要一点时间才能

正确显示。

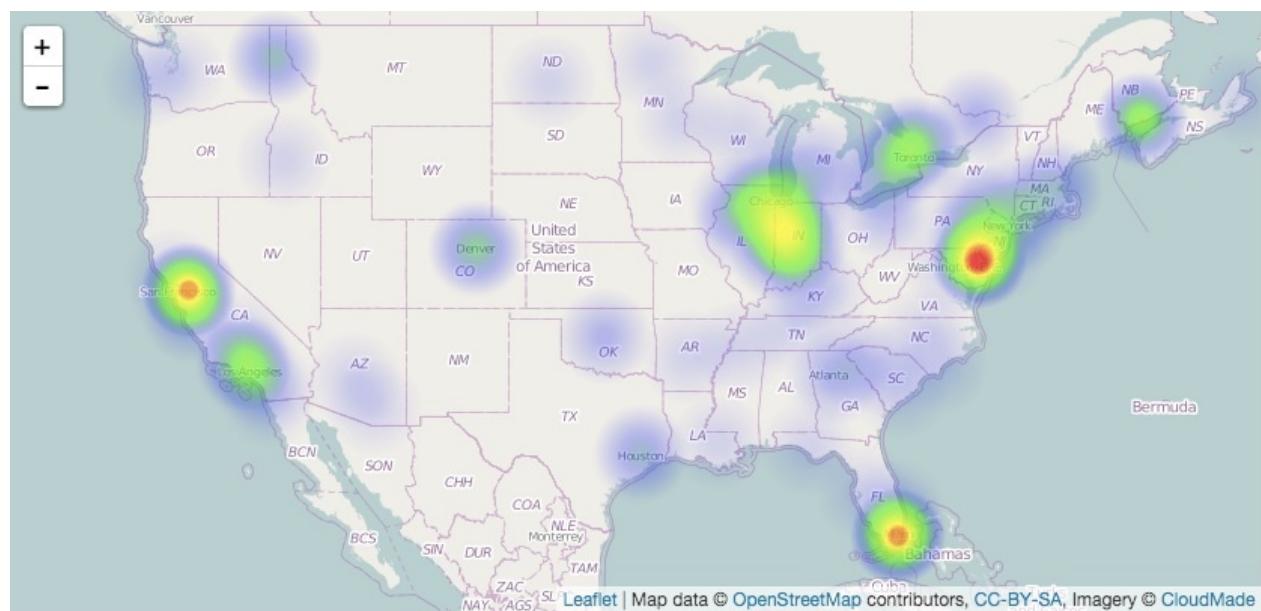
在 bettermap 的配置界面，我提供了 provider，可以选择各种地图提供商。比如中文用户可以选择 GaoDe(高德地图)：



## 其他

leaflet 库有丰富的[插件资源](#)。比如

- [热力图](#)



小贴士：其实 Kibana 官方效果的标记集群也是插件实现的，叫[markercluster](#)

# terms

---

状态：稳定

基于 Elasticsearch 的 terms facet 接口数据展现表格，条带图，或者饼图。

## 参数

---

- field

用于计算 facet 的字段名称。

- script

用于提交 facet 的 scriptField 脚本字符串。系我的 fork 中新增的功能，仅在 fmode 为 script 时生效。

- exclude

要从结果数据中排除掉的 terms

- missing

设为假，就可以不显示数据集内有多少结果没有你指定的字段。

- other

设为假，就可以不显示聚合结果在你的 size 属性设定范围以外的总计数值。

- size

显示多少个 terms

- order

terms 模式可以设置：count, term, reverse\_count 或者 reverse\_term ; terms\_stats 模式可以设置：term, reverse\_term, count, reverse\_count, total, reverse\_total, min, reverse\_min, max, reverse\_max, mean 或者 reverse\_mean

- donut

在饼图(pie)模式，在饼中画个圈，变成甜甜圈样式。

- tilt

在饼图(pie)模式，倾斜饼变成椭圆形。

- labels

在饼图(pie)模式，在饼图分片上绘制标签。

- arrangement

在条带(bar)或者饼图(pie)模式，图例的摆放方向。可以设置：水平(horizontal)或者垂直(vertical)。

- chart

可以设置：table, bar 或者 pie

- counter\_pos

图例相对于图的位置，可以设置：上(above)，下(below)，或者不显示(none)。

- spyable

设为假，不显示审查(inspect)按钮。

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些？可设选项有：all, pinned, unpinned, selected

- queries.ids

如果设为 selected 模式，具体被选的请求编号。

- tmode

Facet 模式：terms 或者 terms\_stats。

- fmode

Field 模式：normal 或者 script。我的 fork 中新增参数，normal 行为和原版一致，选择 script 时，scriptField 参数生效。

- tstat

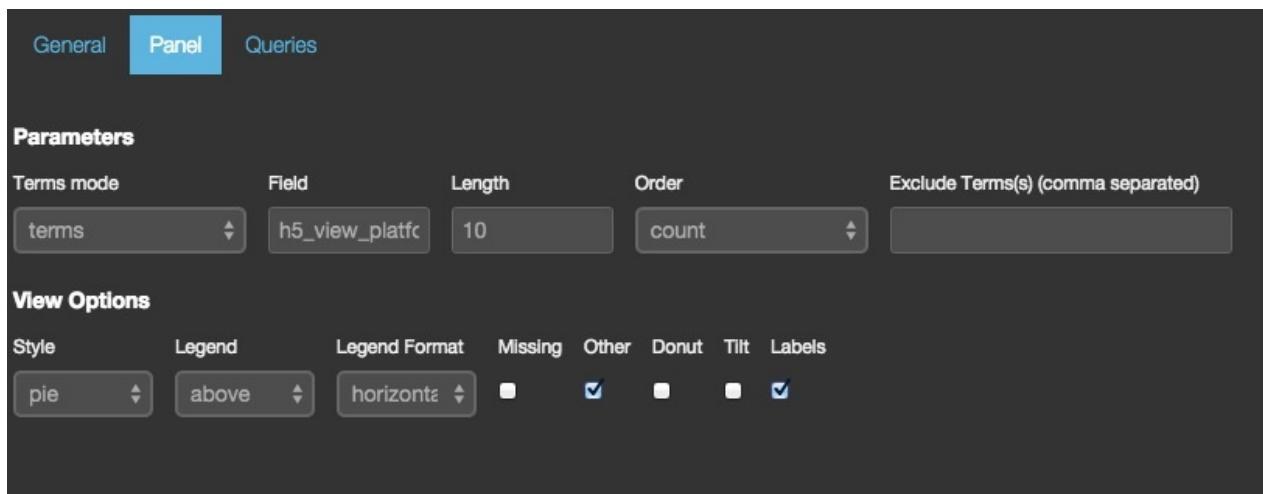
Terms\_stats facet stats 字段。

- valuefield

Terms\_stats facet value 字段。

## 界面配置说明

terms 面板是针对单项数据做聚合统计的面板。可配置项比较简单：



主要分为两部分，数据模式和显示风格。

## 数据模式

terms 面板能够使用两种数据模式(也是 Kibana 大多数面板所使用的)：

- terms

terms 即普通的分类计数(类比为 `group by` 语法)。填写具体字段名即可。此外，排序(`order by`)和结果数(`limit`)也可以定义，具体选项介绍见本页前半部分。

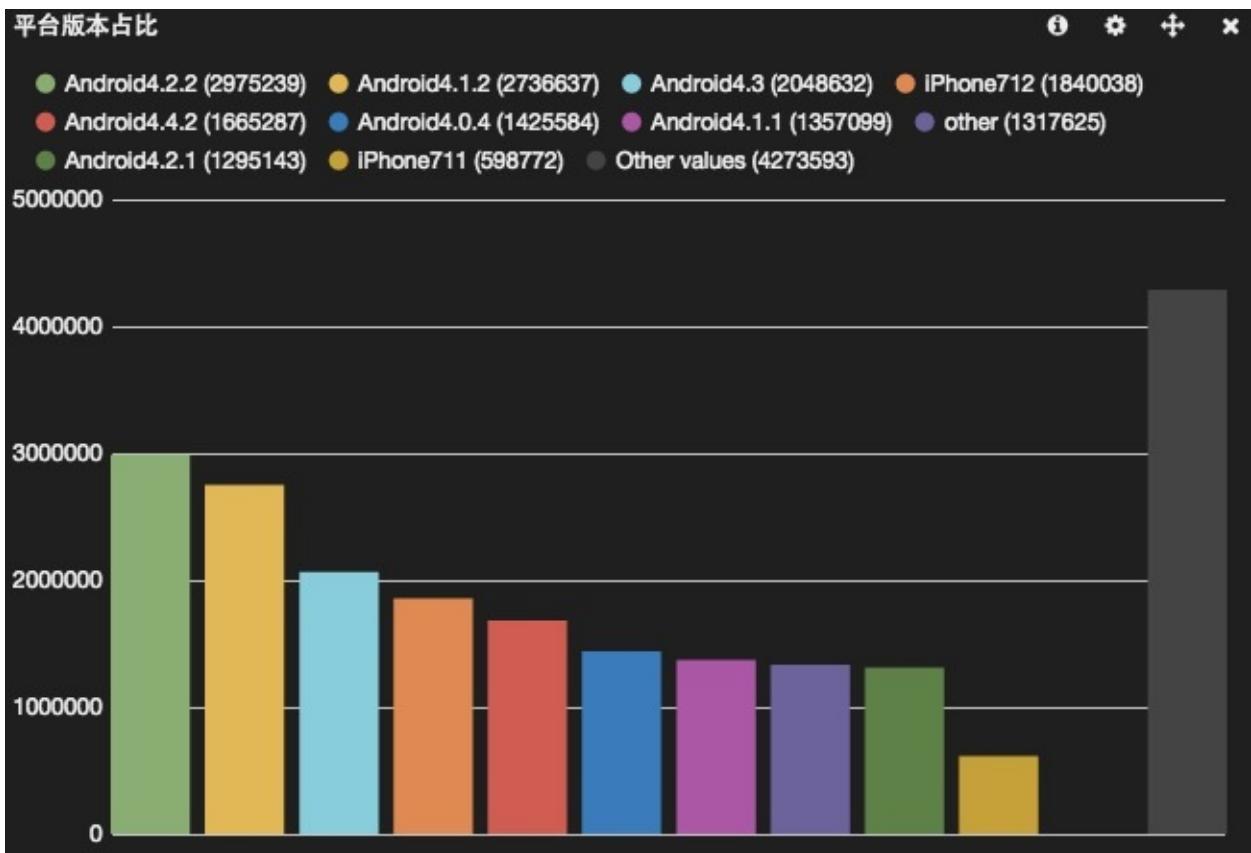
- terms\_stats

terms\_stats 在 terms 的基础上，获取另一个数值类型字段的统计值作为显示内容。可选的统计值有：`count`, `total_count`, `min`, `max`, `total`, `mean`。最常用的就是 `mean`。

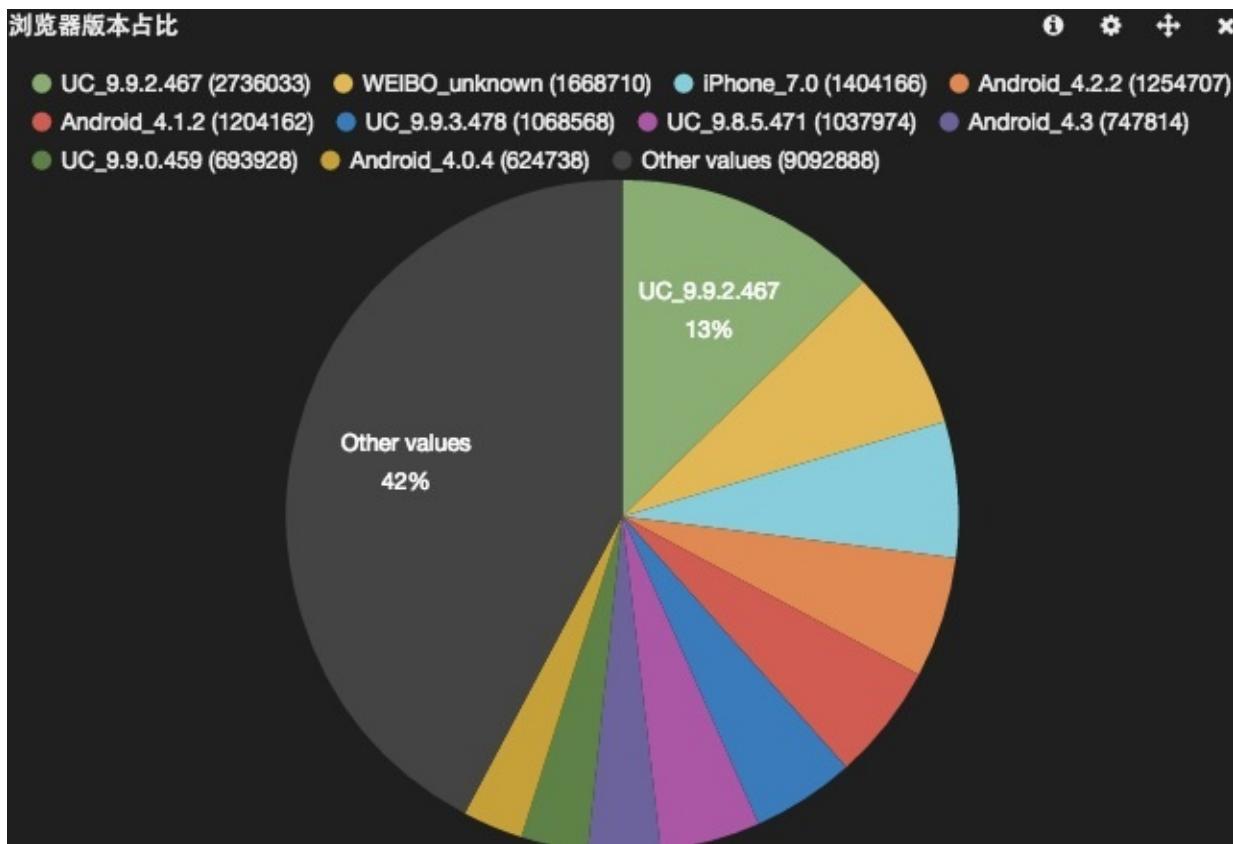
## 显示风格

terms 面板可以使用多个风格来显示数据。

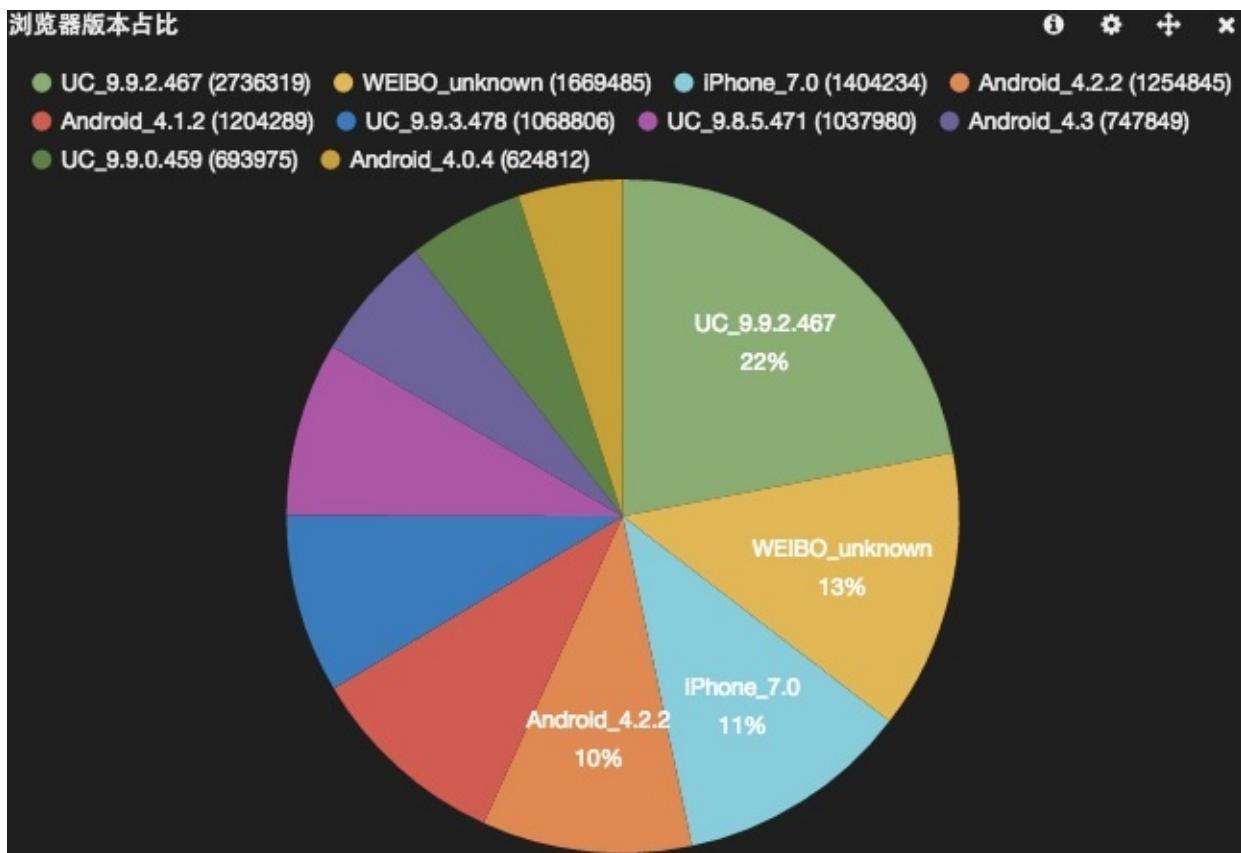
- bar



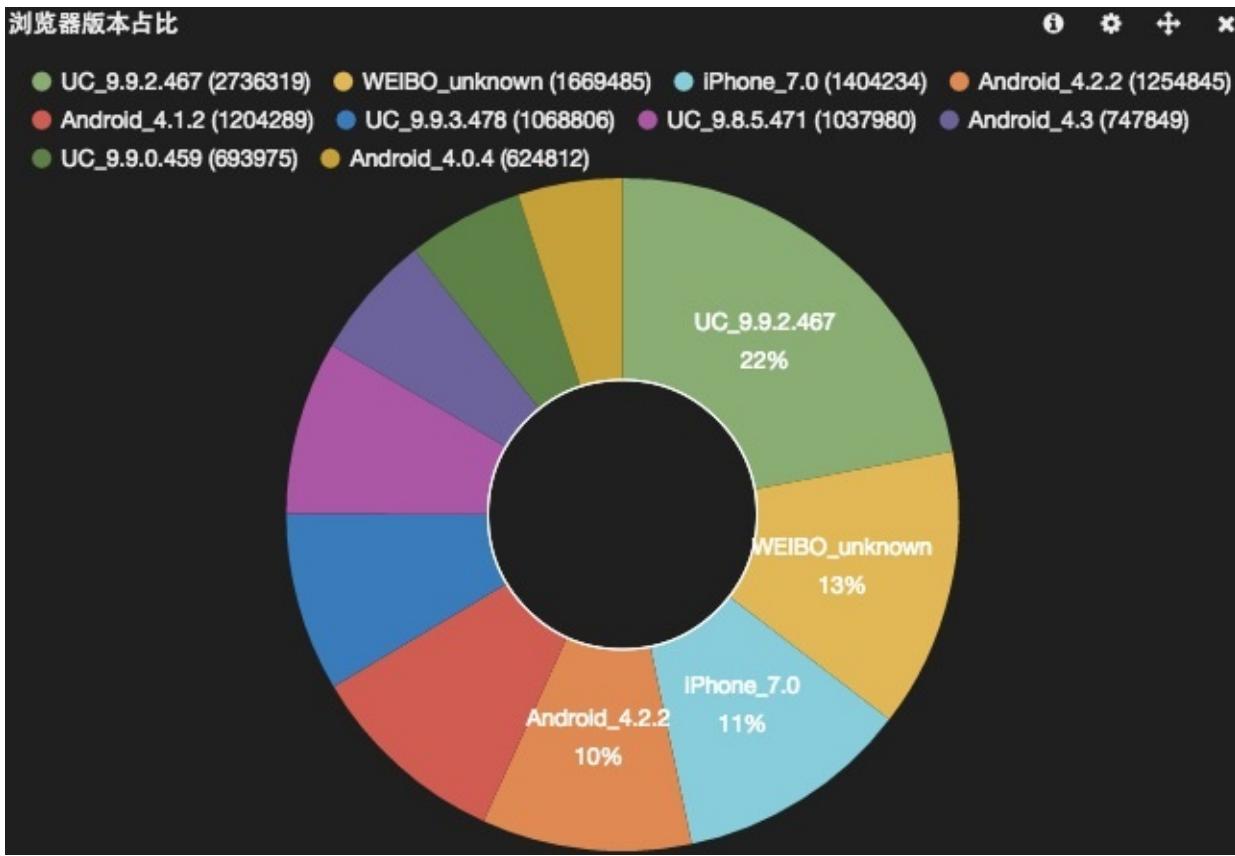
- pie



这时候可能就会觉得这个 "other value" 太大了，又不关心它。那么可以在配置里去掉 other 的勾选。图形会变成这样：



如果勾选 "donut", 则可以看到圈圈饼效果：



- table

如果你是个对数字敏感的人，或者主要数据差距不大，通过 bar 或者 pie 方式不是很明显，那么看表格最好了：

平台版本占比

Term	Count	Action
Android4.2.2	2975523	Q Ø
Android4.1.2	2736628	Q Ø
Android4.3	2048991	Q Ø
iPhone712	1840385	Q Ø
Android4.4.2	1665541	Q Ø
Android4.0.4	1425813	Q Ø
Android4.1.1	1357238	Q Ø
other	1317517	Q Ø
Android4.2.1	1294927	Q Ø
iPhone711	598542	Q Ø
Missing field	0	Q Ø
Other values	4274653	

注意这个表格只有单列数据，使用配置里定义的排序，不像 table 面板。

如果你需要同时看多种统计数据，则应该使用 [stats 面板](#)。

## script field

在 fmode 选择 script 的时候，可以填写 script 脚本字符串获取脚本化字段结果做聚合。

在 scriptField 输入框中输入

```
doc['path.raw'].value
```

的时候，效果完全等价于直接在 Field 输入框中输入

```
path.raw
```

因为 script 和 analyzer 的次序关系，务必使用带有 "**not\_analyzed**" 属性的字段。否则一条数据中只会有一个分词结果参与后续聚合运算。

支持的 script 语法，请参阅 ES 官方文档：[http://www.elasticsearch.org/guide/en/elasticsearch/reference/3.0/modules-scripting.html#\\_document\\_fields](http://www.elasticsearch.org/guide/en/elasticsearch/reference/3.0/modules-scripting.html#_document_fields)

需要注意的是，出于安全考虑，ES 1.4 以下大多建议关闭动态脚本运行的支持；在 1.4 新增了沙箱运行并设置为默认。所以，建议在 ES 1.4 的前提下使用该特性。

# column

状态：稳定

这是一个伪面板。目的是让你在一列中添加多个其他面板。虽然 column 面板状态是稳定，它的限制还是很多的，比如不能拖拽内部的小面板。未来的版本里，column 面板可能被删除。

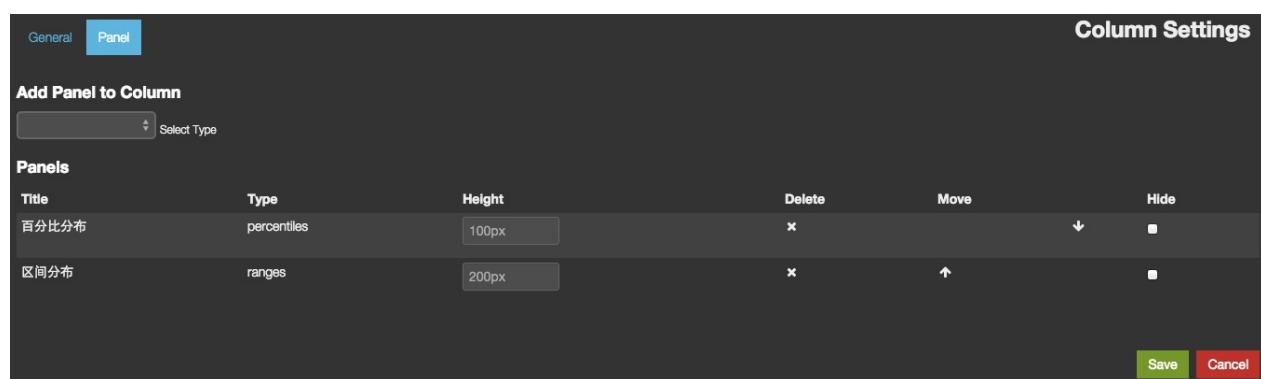
## 参数

- panel

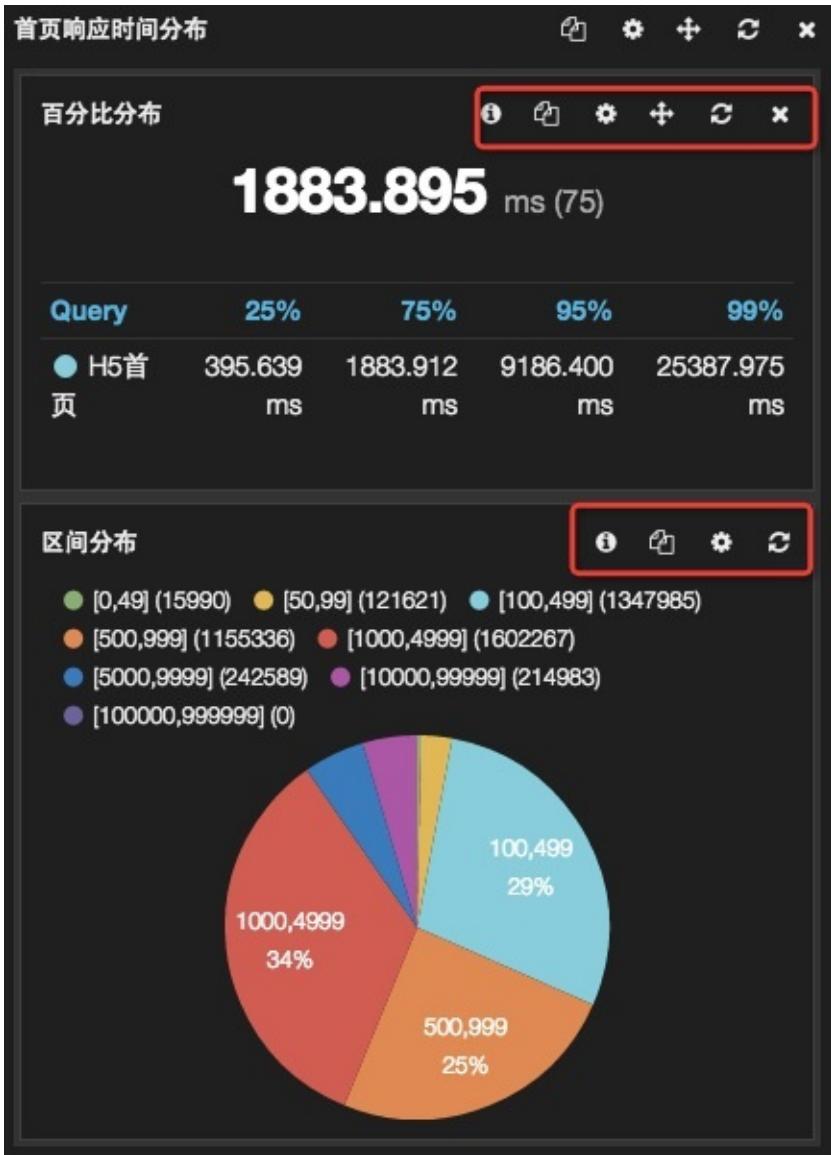
面板对象构成的数组

## 界面配置说明

column 面板是为了在高度较大的 row 中放入多个小 panel 准备的一个容器。其本身配置界面和 row 类似只有一个 panel 列表：



在 column 中的具体的面板本身的设定，需要点击面板自带的配置按钮来配置：



# stats

---

状态: Beta

基于 Elasticsearch 的 statistical Facet 接口实现的统计聚合展示面板。

## 参数

---

- **format**

返回值的格式。默认是 number, 可选值还有 : money, bytes, float。

- **style**

主数字的显示大小, 默认为 24pt。

- **mode**

用来做主数字显示的聚合值, 默认是 count, 可选值为 : count(计数), min(最小值), max(最大值), mean(平均值), total(总数), variance(方差), std\_deviation(标准差), sum\_of\_squares(平方和)。

- **show**

统计表格中具体展示的哪些列。默认为全部展示, 可选列名即 mode 中的可选值。

- **spyable**

设为假, 不显示审查(inspect)按钮。

## 请求

- **请求对象**

这个对象描述本面板使用的请求。

- **queries.mode**

在可用请求中应该用哪些? 可设选项有 : all, pinned, unpinned, selected

- **queries.ids**

如果设为 selected 模式, 具体被选的请求编号。

# query

query 面板和 filter 面板都是特殊类型的面板，在 dashboard 上有且仅有一个。不能删除不能添加。

query 和 filter 的普通样式和基本操作，在官方的[请求和过滤](#)章节已经讲述过。这里，额外讲一下一些高阶功能。

## 请求类型

query 搜索框支持三种请求类型：

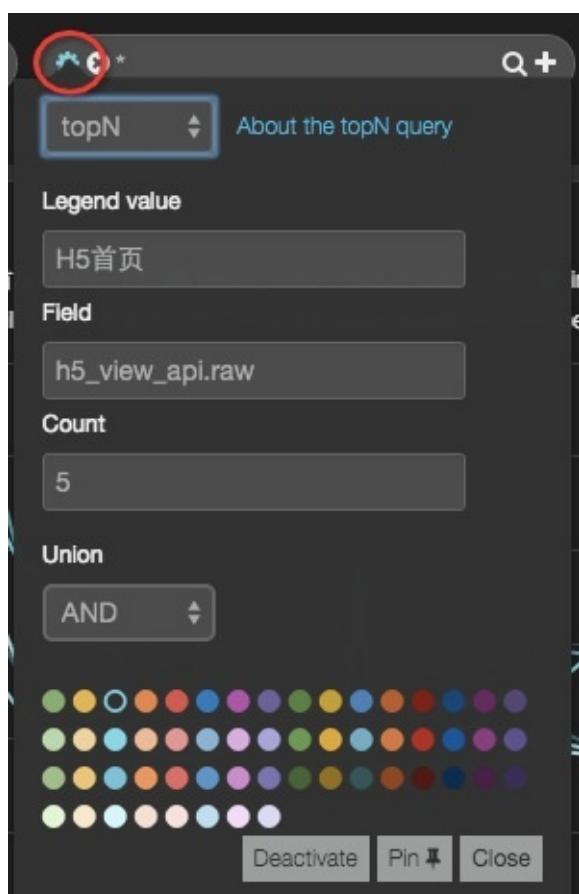
- lucene

这也是默认的类型，使用要点就是请求语法。语法说明

见：<http://www.elasticsearch.org/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax>

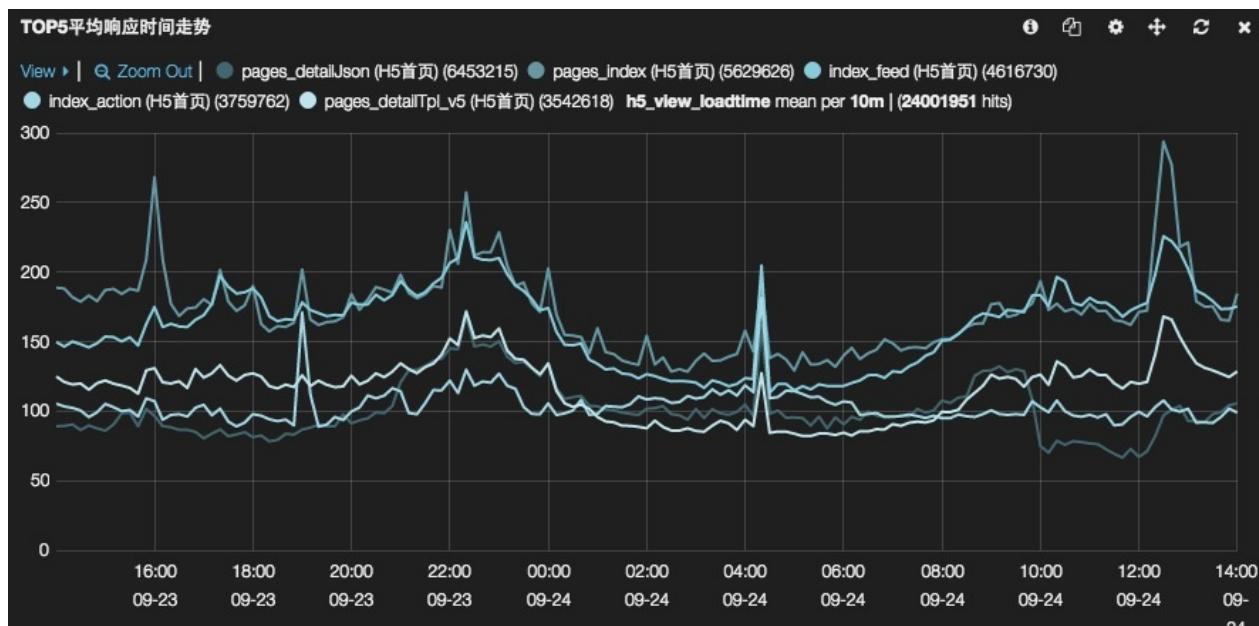
- regex
- topN

topN 是一个方便大家进行多项对比搜索的功能。其配置界面如下：



注意 topN 后颜色选择器的小圆点变成了齿轮状！

其运行实质，是先根据你填写的 field 和 size，发起一次 termsFacet 查询，获取 topN 的 term 结果；然后拿着这个列表，逐一发起附加了 term 条件的其他请求(比如绑定在 histogram 面板就是 date\_histogram 请求，stats 面板就是 termStats 请求)，也就获得了 topN 结果。



小贴士：如果 ES 响应较慢的时候，你甚至可以很明显的看到 histogram 面板上的多条曲线是一条一条出来数据绘制的。

# trends

---

状态: Beta

以证券报价器风格展示请求随着时间移动的情况。比如说：当前时间是 1:10pm，你的时间选择器设置的是 "Last 10m"，而本面板的 "Time Ago" 参数设置的是 "1h"，那么面板会显示的是请求结果从 12:00-12:10pm 以来变化了多少。

## 参数

---

- ago

描述需要对比请求的时期的时间数值型字符串。

- arrangement

'horizontal' 或 'vertical'

- spyable

设为假，不显示审查(inspect)按钮。

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些？可设选项有： all, pinned, unpinned, selected

- queries.ids

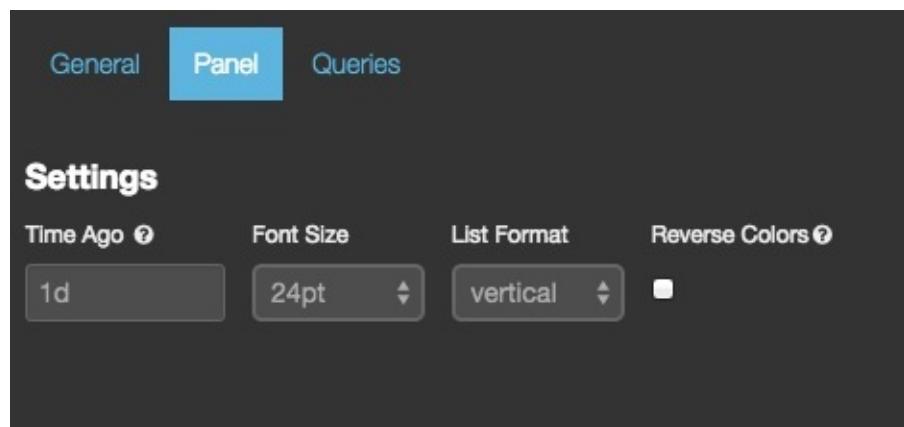
如果没为 selected 模式，具体被选的请求编号。

---

## 界面配置说明

---

trends 面板用来对比实时数据与过去某天的同期数据的量的变化。配置很简单，就是设置具体某天前：



效果如下：

TOP5请求量今日涨幅

① ⌂ ⚙ ⌂ ✕

- ▼**-8.16%** (pages\_detailJson (H5首页))
- ▼**-0.23%** (pages\_index (H5首页))
- ▲**2.19%** (index\_feed (H5首页))
- ▲**6.05%** (index\_action (H5首页))
- ▲**9.25%** (pages\_detailTpl\_v5 (H5首页))

# 文本(text)

---

状态：稳定

文本面板用来显示静态文本内容，支持 markdown，简单的 html 和纯文本格式。

## 参数

---

- mode

'html', 'markdown' 或者 'text'

- content

面板内容，用 mode 参数指定的标记语言书写

# sparklines

状态: 试验性

sparklines 面板显示微型时间图。目的不是显示一个确切的数值，而是以紧凑的方式显示时间序列的形态。

## 参数

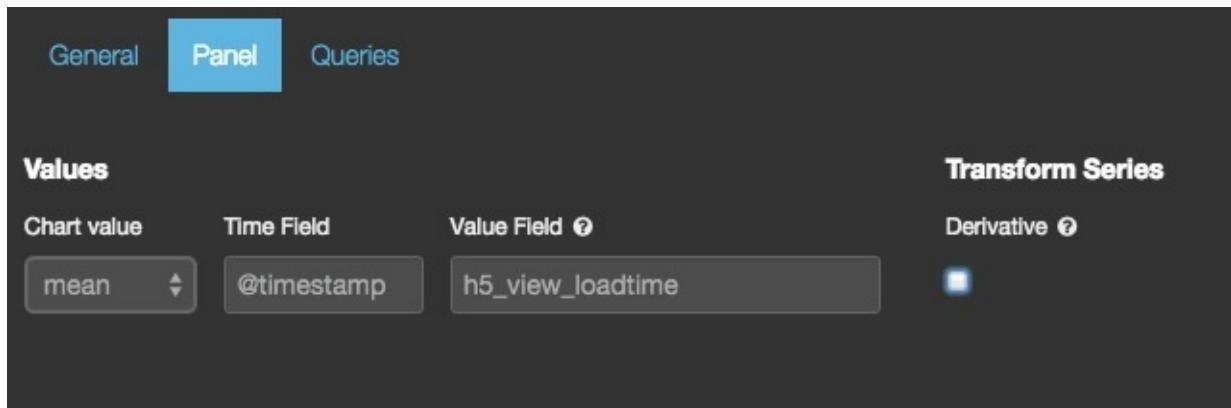
- mode 用作 Y 轴的数值模式。除 count 以外，都需要定义 `value_field` 字段。可选值有：count, mean, max, min, total.
- time\_field X 轴字段。必须是 Elasticsearch 中的时间类型字段。
- value\_field 如果 mode 设置为 mean, max, min 或者 total, Y 轴字段。必须是数值类型字段。
- interval 如果有现成的时间过滤器，Sparkline 会自动计算间隔。如果没有，就用这个间隔。默认是 5 分钟。
- spyable 显示 inspect 图标。

### 请求(queries)

- 请求对象 这个对象描述本面板使用的请求。
  - queries.mode 在可用请求中应该用哪些？可设选项有：all, pinned, unpinned, selected
  - queries.ids 如果设为 selected 模式，具体被选的请求编号。

## 界面配置说明

sparklines 面板其实就是 histogram 面板的缩略图模式。在配置上，只能选择 Chart value 模式，填写 Time Field 或者 Value Field 字段。上文描述中的 interval 在配置页面上是看不到的。

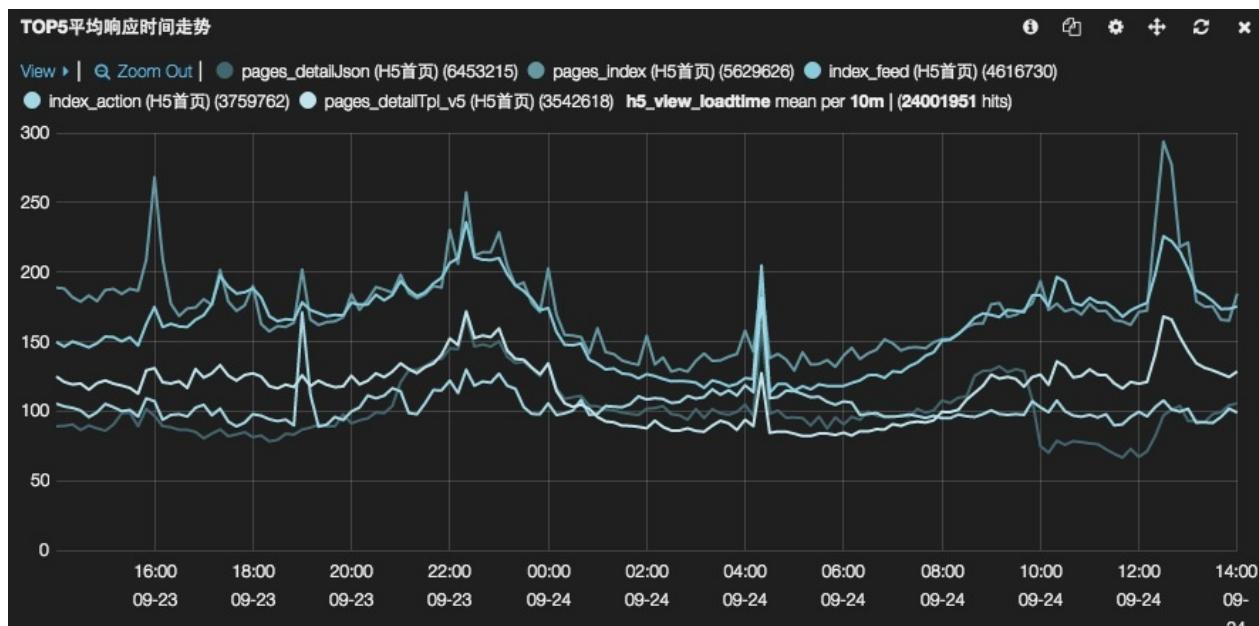


我们可以对比一下对同一个 topN 请求绘制的 sparklines 和 histogram 面板的效果：

- sparklines



- histogram



注 : topN 请求的配置和说明, 见 [query 面板](#)

# hits

---

状态: 稳定

hits 面板显示仪表板上每个请求的 hits 数, 具体的显示格式可以通过 "chart" 属性配置指定。

## 参数

---

- arrangement

在条带(bar)或者饼图(pie)模式, 图例的摆放方向。可以设置: 水平(horizontal)或者垂直(vertical)。

- chart

可以设置: none, bar 或者 pie

- counter\_pos

图例相对于图的位置, 可以设置: 上(above), 下(below)

- donut

在饼图(pie)模式, 在饼中画个圈, 变成甜甜圈样式。

- tilt

在饼图(pie)模式, 倾斜饼变成椭圆形。

- labels

在饼图(pie)模式, 在饼图分片上绘制标签。

- spyable

设为假, 不显示审查(inspect)图标。

## 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些? 可设选项有: all, pinned, unpinned, selected

- queries.ids

如果设为 selected 模式, 具体被选的请求编号。

# goal

---

状态: 稳定

goal 面板在一个饼图上显示到达指定目标的进度。

## 参数

---

- donut

在饼图(pie)模式, 在饼中画个圈, 变成甜甜圈样式。

- tilt

在饼图(pie)模式, 倾斜饼变成椭圆形。

- legend

图例的位置, 上、下或者无。

- labels

在饼图(pie)模式, 在饼图分片上绘制标签。

- spyable

设为假, 不显示审查(inspect)图标。

## 请求(queries)

- 请求对象

- query.goal

goal 模式的指定目标

# percentile

---

状态: Beta

基于 Elasticsearch 的 percentile Aggregation 接口实现的统计聚合展示面板。

## 参数

---

- **format**

返回值的格式。默认是 number, 可选值还有 : money, bytes, float。

- **style**

主数字的显示大小, 默认为 24pt。

- **modes**

用来做百分比的聚合值。包括25%, 50%, 75%, 90%, 95%, 99%。

- **show**

统计表格中具体展示的哪些列。默认为全部展示, 可选列名即 modes 中的可选值。

- **spyable**

设为假, 不显示审查(inspect)按钮。

## 请求

- **请求对象**

这个对象描述本面板使用的请求。

- **queries.mode**

在可用请求中应该用哪些? 可设选项有 : all, pinned, unpinned, selected

- **queries.ids**

如果设为 selected 模式, 具体被选的请求编号。

---

## 界面配置说明

---

percentile 面板界面与 stats 面板界面类似。



## 代码实现要点

- percentile Aggregation 是 Elasticsearch 从 1.1.0 开始新加入的实验性功能，而且在 1.3.0 之后其返回的数据结构发生了变动。所以代码中对 ESversion 要做判断和兼容性处理。
- percentile Aggregation 返回的数据中，强制保留了百分数的小数点后一位，这导致在 js 处理中会把小数点当做是属性调用的操作符。所以需要在前端展示的 "." 替换成后端使用的 "\_"。
- percentile Aggregation 请求中，不支持使用中文做 aggregation name。如果 `query.alias` 写了中文的，就会出问题。所以这里直接采用序号了。

# range

---

基于 Elasticsearch 的 range facet 接口数据展现表格，条带图，或者饼图。

## 参数

---

- field

用于计算 facet 的字段名称。

- values

用于计算 facet 的数值范围数组。数组每个元素包括：

- from

range 范围的起始点

- to

range 范围的结束点

- exclude

要从结果数据中排除掉的 terms

- missing

设为假，就可以不显示数据集内有多少结果没有你指定的字段。

- other

设为假，就可以不显示聚合结果在你的 size 属性设定范围以外的总计数值。

- size

显示多少个 terms

- order

terms 模式可以设置：count, term, reverse\_count 或者 reverse\_term ; terms\_stats 模式可以设置：term, reverse\_term, count, reverse\_count, total, reverse\_total, min, reverse\_min, max, reverse\_max, mean 或者 reverse\_mean

- donut

在饼图(pie)模式，在饼中画个圈，变成甜甜圈样式。

- tilt

在饼图(pie)模式，倾斜饼变成椭圆形。

- labels

在饼图(pie)模式，在饼图分片上绘制标签。

- arrangement

在条带(bar)或者饼图(pie)模式，图例的摆放方向。可以设置：水平(horizontal)或者垂直(vertical)。

- chart

可以设置：table, bar 或者 pie

- counter\_pos

图例相对于图的位置，可以设置：上(above)，下(below)，或者不显示(none)。

- spyable

设为假，不显示审查(inspect)按钮。

### 请求(queries)

- 请求对象

这个对象描述本面板使用的请求。

- queries.mode

在可用请求中应该用哪些？可设选项有：all, pinned, unpinned, selected

- queries.ids

如果设为 selected 模式，具体被选的请求编号。

---

## 界面配置说明

range 面板是针对单项数据做聚合统计的面板。效果与 terms 面板类似。其配置界面如下：

## Parameters

Ranges mode

ranges

Field

h5\_view\_loadt

From	To	Delete
0	99	✖
100	999	✖
1000	9999	✖
10000	99999	✖
100000	999999	✖
1000000	9999999	✖

 Add value

## 认证鉴权

---

# 配置Kibana的CAS验证

感谢携程网的[@childe]童鞋贡献本节内容

我们公司用的是 CAS 单点登陆, 用如下工具将kibana集成到此单点登陆系统

## 准备工具

- nginx: 仅仅是为了记录日志, 不用也行
- nodejs: 为了跑 kibana-authentication-proxy
- kibana: <https://github.com/elasticsearch/kibana>
- kibana-authentication-proxy: <https://github.com/fangli/kibana-authentication-proxy>

## 配置

1. nginx 配置 8080 端口, 反向代理到 es 的 9200
2. git clone kibana-authentication-proxy
3. git clone kibana
4. 将 kibana 软链接到 kibana-authentication-proxy 目录下
5. 配置 kibana-authentication-proxy/config.js

可能有如下参数需要调整:

```
es_host      #这里是nginx地址
es_port      #nginx的8080
listen_port   #node的监听端口, 80
listen_host    #node的绑定IP, 可以0.0.0.0
cas_server_url #CAS地址
```

6. 安装 kibana-authentication-proxy 的依赖, `npm install express`, 等
7. 运行 `node kibana-authentication-proxy/app.js`

## 原理

- app.js 里面 `app.get('/config.js', kibana3configjs)`; 返回了一个新的 config.js, 不是用的 kibana/config.js, 在这个配置里面, 调用 ES 数据的 URL 前面加了一个 `_es` 的前缀
- 在 app.js 入口这里, 有两个关键的中间层(我也不知道叫什么)被注册: 一个是 `configureCas`, 一个是 `configureESProxy`
- 一个请求来的时候, 会到 `configureCas` 判断是不是已经登陆到 CAS, 没有的话就转到 cas 登陆页面
- `configureESProxy` 在 lib/es-proxy.js 里, 会把 `_es` 打头的请求(其实就是请求 es 数据的请求)转发到真正的 es 接口那里(我们这里是 nginx)

## 请求路径

```
node(80) <=> nginx(8080) <=> es(9200)
```

kibana-authentication-proxy 本身没有记录日志的代码, 而且转发 es 请求用的流式的(看起来), 并不能记录详细的 request body. 所以我们就用 nginx 又代理一层做日志了..

# Auth WebUI in Mojolicious

社区已经有用 nodejs 或者 rubyonrails 写的 kibana-auth 方案了。不过我这两种语言都不太擅长，只会写一点点 Perl5 代码，所以我选择用 [Mojolicious](#) web 开发框架来实现我自己的 kibana 认证鉴权。

整套方案的代码以 `kbnauth` 子目录形式存在于我的 [kibana 仓库](#) 中，如果你不想用这套认证方案，照旧使用 `src` 子目录即可。事实上，`kbnauth/public/` 目录下的静态文件我都是通过软连接方式指到 `src/` 下的。

## 特性

- 全局透明代理

和 nodejs 实现的那套方案不同，我这里并没有使用 `_es/` 这样附加的路径。所有发往 Elasticsearch 的请求都是通过这个方案来控制。除了使用 `config.js.ep` 模版来定制 `elasticsearch` 地址设置以外，方案还会伪造 `/_nodes` 请求的响应体，伪造的响应体中永远只有运行着认证方案的这台服务器的 IP 地址。

这么做的原因是我的 `kibana` 升级了 `elasticjs` 版本，新版本默认会通过这个 API 获取 `nodes` 列表，然后浏览器直接轮询多个 IP 获得响应。

注意：`Mojolicious` 有一个环境变量叫 `max_message_size`，默认是 **10MB**，即只允许代理响应大小在 **10MB** 以内的数据。我在 `script/kbnauth` 启动脚本中把它修改成了 **0**，即不限制。如果你有这方面的需求，可以改成任意你想要的阈值。

- 使用 `kibana-auth` elasticsearch 索引做鉴权

因为所有的请求都会发往代理服务器(即运行着认证鉴权方案的服务器)，每个用户都可以有自己的仪表板空间(没错，这招是从 `kibana-proxy` 项目学来的，每个用户使用单独的 `kibana-int-$username` 索引保存自己的仪表板设置)。而本方案还提供另一个高级功能：还可以通过另一个新的索引 `kibana-auth` 来指定每个用户所能访问的 Elasticsearch 集群地址和索引列表。

给用户 "sri" 添加鉴权信息的命令如下：

```
$ curl -XPOST http://127.0.0.1:9200/kibana-auth/indices/sri -d '{  
  "prefix": ["logstash-sri", "logstash-ops"],  
  "server": "192.168.0.2:9200"  
}'
```

这就意味着用户 "sri" 能访问的，是存在 "192.168.0.2:9200" 上的 "logstash-sri-YYYY.MM.dd" 或者 "logstash-ops-YYYY.MM.dd" 索引。

小贴士：所以你在 `kbn_auth.conf` 里配置的 `eshost/esport`，其实并不意味着 `kibana` 数据的来源，而是认证方案用来请求 `kibana-auth` 信息的地址！

- 使用 [Authen::Simple](#) 框架做认证

`Authen::Simple` 是一个很棒的认证框架，支持非常多的认证方法。比如：LDAP, DBI, SSH, Kerberos, PAM, SMB, NIS, PAM, ActiveDirectory 等。

默认使用的是 `Passwd` 方法。也就是用 `htpasswd` 命令行在本地生成一个 `.htpasswd` 文件存用户名密码。

如果要使用其他方法，比如用 LDAP 认证，只需要配置 `kbn_auth.conf` 文件就行了：

```
authen => {  
  LDAP => {  
    host  => 'ad.company.com',  
    binddn => 'proxyuser@company.com',  
    bindpw => 'secret',
```

```
        basedn => 'cn=users,dc=company,dc=com',
        filter => '(&(objectClass=organizationalPerson)(objectClass=user)(sAMAccountName=%s))'
    },
}
```

可以同时使用多种认证方式，但请确保每种都是有效可用的。某一个认证服务器连接超时也会影响到其他认证方式超时。

## 安装

该方案代码只有两个依赖：Mojolicious 框架和 Authen::Simple 框架。我们可以通过 cpanm 部署：

```
curl http://xrl.us/cpanm -o /usr/local/bin/cpanm
chmod +x /usr/local/bin/cpanm
cpanm Mojolicious Authen::Simple::Passwd
```

如果你需要使用其他认证方法，每个方法都需要另外单独安装。比如使用 LDAP 部署，就再运行一行：`cpanm Authen::Simple::LDAP` 就可以了。

小贴士：如果你是在一个新 RHEL 系统上初次运行代码，你可能会发现有报错说找不到 `Digest::SHA` 模块。这个模块其实是 Perl 核心模块，但是 RedHat 公司把所有的 Perl 核心模块单独打包成了 `perl-core.rpm`，所以你得先运行一下 `yum install -y perl-core` 才行。我讨厌 RedHat！

## 运行

```
cd kbnauth
# 开发环境监听 3000 端口，使用单进程的 morbo 服务器调试
morbo script/kbnauth
# 生产环境监听 80 端口，使用高性能的 hypnotoad 服务器，具体端口在 kbn_auth.conf 中定义
hypnotoad script/kbnauth
```

现在，打开浏览器，就可以通过默认的用户名/密码："sri/secr3t" 登录进去了。(sri 是 Mojolicious 框架的作者，感谢他为 Perl5 社区提供这么高效的 web 开发框架)

注意：这时候你虽然认证通过进去了 kibana 页面，但是还没有赋权。按照上面提到的 `kibana-auth` 命令操作，才算全部完成。

# 源码剖析与二次开发

---

Kibana 3 作为 ELKstack 风靡世界的最大推动力，其与优美的界面配套的简洁的代码同样功不可没。事实上，graphite 社区就通过移植 kibana 3 代码框架的方式，启动了 [grafana 项目](#)。至今你还能在 grafana 源码找到二十多处 "kbn" 字样。

巧合的是，在 Kibana 重构 v4 版的同时，grafana 的 v2 版也到了 Alpha 阶段，从目前的预览效果看，主体 *dashboard* 沿用了 Kibana 3 的风格，不过添加了额外的菜单栏，供用户权限设置等使用——这意味着 grafana 2 跟 kibana 4 一样需要一个单独的 server 端。

笔者并非专业的前端工程师，对 angularjs 也处于一本入门指南都没看过的水准。所以本节内容，只会抽取一些个人经验中会有涉及到的地方提出一些"私货"。欢迎方家指正。

# 源码目录结构

下面是 kibana 源码的全部文件的 tree 图：

```
.  
├── app  
│   ├── app.js  
│   ├── components  
│   │   ├── extend-jquery.js  
│   │   ├── kbn.js  
│   │   ├── lodash.extended.js  
│   │   ├── require.config.js  
│   │   └── settings.js  
│   ├── controllers  
│   │   ├── all.js  
│   │   ├── dash.js  
│   │   ├── dashLoader.js  
│   │   ├── pulldown.js  
│   │   └── row.js  
│   ├── dashboards  
│   │   ├── blank.json  
│   │   ├── default.json  
│   │   ├── guided.json  
│   │   ├── logstash.js  
│   │   ├── logstash.json  
│   │   ├── noted.json  
│   │   ├── panel.js  
│   │   └── test.json  
│   ├── directives  
│   │   ├── addPanel.js  
│   │   ├── all.js  
│   │   ├── arrayJoin.js  
│   │   ├── configModal.js  
│   │   ├── confirmClick.js  
│   │   ├── dashUpload.js  
│   │   ├── esVersion.js  
│   │   ├── kibanaPanel.js  
│   │   ├── kibanaSimplePanel.js  
│   │   ├── ngBlur.js  
│   │   ├── ngModelOnBlur.js  
│   │   ├── resizable.js  
│   │   └── tip.js  
│   ├── factories  
│   │   └── store.js  
│   ├── filters  
│   │   └── all.js  
│   ├── panels  
│   │   ├── bettermap  
│   │   │   ├── editor.html  
│   │   ├── leaflet  
│   │   │   ├── images  
│   │   │   │   ├── layers-2x.png  
│   │   │   │   ├── layers.png  
│   │   │   │   ├── marker-icon-2x.png  
│   │   │   │   ├── marker-icon.png  
│   │   │   │   └── marker-shadow.png  
│   │   │   ├── leaflet-src.js  
│   │   │   ├── leaflet.css  
│   │   │   ├── leaflet.ie.css  
│   │   │   ├── leaflet.js  
│   │   │   ├── plugins.css  
│   │   │   ├── plugins.js  
│   │   │   └── providers.js  
│   │   ├── module.css  
│   │   ├── module.html  
│   │   └── module.js  
│   ├── column  
│   │   ├── editor.html  
│   │   ├── module.html  
│   │   ├── module.js  
│   │   └── panelgeneral.html  
│   ├── dashcontrol  
│   │   ├── editor.html  
│   │   ├── module.html  
│   │   └── module.js  
│   └── derivequeries
```

```
    └── editor.html
    └── module.html
    └── module.js
  └── fields
    └── editor.html
    └── micropanel.html
    └── module.html
    └── module.js
  └── filtering
    └── editor.html
    └── meta.html
    └── module.html
    └── module.js
  └── force
    └── editor.html
    └── module.html
    └── module.js
  └── goal
    └── editor.html
    └── module.html
    └── module.js
  └── histogram
    └── editor.html
    └── interval.js
    └── module.html
    └── module.js
    └── queriesEditor.html
    └── styleEditor.html
    └── timeSeries.js
  └── hits
    └── editor.html
    └── module.html
    └── module.js
  └── map
    └── editor.html
    └── lib
      └── jquery.jvectormap.min.js
      └── map.cn.js
      └── map.europe.js
      └── map.usa.js
      └── map.world.js
    └── module.html
    └── module.js
  └── multifieldhistogram
    └── editor.html
    └── interval.js
    └── markersEditor.html
    └── meta.html
    └── module.html
    └── module.js
    └── styleEditor.html
    └── timeSeries.js
  └── percentiles
    └── editor.html
    └── module.html
    └── module.js
  └── query
    └── editor.html
    └── editors
      └── lucene.html
      └── regex.html
      └── topN.html
    └── help
      └── lucene.html
      └── regex.html
      └── topN.html
    └── helpModal.html
    └── meta.html
    └── module.html
    └── module.js
    └── query.css
  └── ranges
    └── editor.html
    └── module.html
    └── module.js
  └── sparklines
    └── editor.html
    └── interval.js
    └── module.html
    └── module.js
    └── timeSeries.js
  └── statisticstrend
```

```
    |   |   |   |   |   |   editor.html
    |   |   |   |   |   |   module.html
    |   |   |   |   |   |   module.js
    |   |   |   |   stats
    |   |   |   |   |   editor.html
    |   |   |   |   |   module.html
    |   |   |   |   |   module.js
    |   |   |   |   table
    |   |   |   |   |   editor.html
    |   |   |   |   |   export.html
    |   |   |   |   |   micropanel.html
    |   |   |   |   |   modal.html
    |   |   |   |   |   module.html
    |   |   |   |   |   module.js
    |   |   |   |   |   pagination.html
    |   |   |   |   terms
    |   |   |   |   |   editor.html
    |   |   |   |   |   module.html
    |   |   |   |   |   module.js
    |   |   |   |   text
    |   |   |   |   |   editor.html
    |   |   |   |   |   lib
    |   |   |   |   |   |   showdown.js
    |   |   |   |   |   module.html
    |   |   |   |   |   module.js
    |   |   |   timepicker
    |   |   |   |   custom.html
    |   |   |   |   editor.html
    |   |   |   |   module.html
    |   |   |   |   module.js
    |   |   |   |   refreshctrl.html
    |   |   |   trends
    |   |   |   |   editor.html
    |   |   |   |   module.html
    |   |   |   |   module.js
    |   |   |   valuehistogram
    |   |   |   |   editor.html
    |   |   |   |   module.html
    |   |   |   |   module.js
    |   |   |   |   queriesEditor.html
    |   |   |   |   styleEditor.html
    |   |   partials
    |   |   |   connectionFailed.html
    |   |   |   dashLoader.html
    |   |   |   dashLoaderShare.html
    |   |   |   dashboard.html
    |   |   |   dasheditor.html
    |   |   |   inspector.html
    |   |   |   load.html
    |   |   |   modal.html
    |   |   |   paneladd.html
    |   |   |   paneleditor.html
    |   |   |   panelgeneral.html
    |   |   |   querySelect.html
    |   |   |   roweditor.html
    |   |   services
    |   |   |   alertSrv.js
    |   |   |   all.js
    |   |   |   dashboard.js
    |   |   |   esVersion.js
    |   |   |   fields.js
    |   |   |   filterSrv.js
    |   |   |   kbnIndex.js
    |   |   |   monitor.js
    |   |   |   panelMove.js
    |   |   |   querySrv.js
    |   |   |   timer.js
    |   config.js
    |   css
    |   |   angular-multi-select.css
    |   |   animate.min.css
    |   |   bootstrap-responsive.min.css
    |   |   bootstrap.dark.min.css
    |   |   bootstrap.light.min.css
    |   |   font-awesome.min.css
    |   |   jquery-ui.css
    |   |   jquery.multiselect.css
    |   |   normalize.min.css
    |   |   timepicker.css
    |   favicon.ico
    |   font
    |   |   FontAwesome.otf
```

```
|   ├── fontawesome-webfont.eot
|   ├── fontawesome-webfont.svg
|   ├── fontawesome-webfont.ttf
|   └── fontawesome-webfont.woff
|
|   ├── img
|   |   ├── annotation-icon.png
|   |   ├── cubes.png
|   |   ├── glyphicons-halflings-white.png
|   |   ├── glyphicons-halflings.png
|   |   ├── kibana.png
|   |   ├── light.png
|   |   ├── load.gif
|   |   ├── load_big.gif
|   |   ├── small.png
|   |   └── ui-icons_222222_256x240.png
|
|   └── index.html
|
└── vendor
    ├── LICENSE.json
    ├── angular
    |   ├── angular-animate.js
    |   ├── angular-cookies.js
    |   ├── angular-dragdrop.js
    |   ├── angular-loader.js
    |   ├── angular-resource.js
    |   ├── angular-route.js
    |   ├── angular-sanitize.js
    |   ├── angular-scenario.js
    |   ├── angular-strap.js
    |   ├── angular.js
    |   ├── bindonce.js
    |   ├── datepicker.js
    |   └── timepicker.js
    ├── blob.js
    ├── bootstrap
    |   ├── bootstrap.js
    |   └── less
    |       ├── accordion.less
    |       ├── alerts.less
    |       ├── bak
    |       |   ├── bootswatch.dark.less
    |       |   └── variables.dark.less
    |       ├── bootstrap.dark.less
    |       ├── bootstrap.less
    |       ├── bootstrap.light.less
    |       ├── bootswatch.dark.less
    |       ├── bootswatch.light.less
    |       ├── breadcrumbs.less
    |       ├── button-groups.less
    |       ├── buttons.less
    |       ├── carousel.less
    |       ├── close.less
    |       ├── code.less
    |       ├── component-animations.less
    |       ├── dropdowns.less
    |       ├── forms.less
    |       ├── grid.less
    |       ├── hero-unit.less
    |       ├── labels-badges.less
    |       ├── layouts.less
    |       ├── media.less
    |       ├── mixins.less
    |       ├── modals.less
    |       ├── navbar.less
    |       ├── navs.less
    |       ├── overrides.less
    |       ├── pager.less
    |       ├── pagination.less
    |       ├── popovers.less
    |       ├── progress-bars.less
    |       ├── reset.less
    |       ├── responsive-1200px-min.less
    |       ├── responsive-767px-max.less
    |       ├── responsive-768px-979px.less
    |       ├── responsive-navbar.less
    |       ├── responsive-utilities.less
    |       ├── responsive.less
    |       ├── scaffolding.less
    |       ├── sprites.less
    |       ├── tables.less
    |       ├── tests
    |           ├── buttons.html
    |           └── css-tests.css
```

```
|   |   ├── css-tests.html
|   |   ├── forms-responsive.html
|   |   ├── forms.html
|   |   ├── navbar-fixed-top.html
|   |   ├── navbar-static-top.html
|   |   └── navbar.html
|   ├── thumbnails.less
|   ├── tooltip.less
|   ├── type.less
|   ├── utilities.less
|   ├── variables.dark.less
|   ├── variables.less
|   ├── variables.light.less
|   └── wells.less
|   ├── chromath.js
|   ├── elasticjs
|   |   ├── elastic-angular-client.js
|   |   └── elastic.js
|   ├── elasticsearch.angular.js
|   ├── filesaver.js
|   ├── jquery
|   |   ├── jquery-1.8.0.js
|   |   ├── jquery-ui-1.10.3.js
|   |   ├── jquery.flot.byte.js
|   |   ├── jquery.flot.events.js
|   |   ├── jquery.flot.js
|   |   ├── jquery.flot.pie.js
|   |   ├── jquery.flot.selection.js
|   |   ├── jquery.flot.stack.js
|   |   ├── jquery.flot.stackpercent.js
|   |   ├── jquery.flot.threshold.js
|   |   ├── jquery.flot.time.js
|   |   ├── jquery.multiselect.filter.js
|   |   └── jquery.multiselect.js
|   ├── jsonpath.js
|   ├── lodash.js
|   ├── modernizr-2.6.1.js
|   ├── moment.js
|   ├── numeral.js
|   ├── require
|   |   ├── css-build.js
|   |   ├── css.js
|   |   ├── require.js
|   |   ├── text.js
|   |   └── tmpl.js
|   ├── simple_statistics.js
|   ├── timezone.js
|   └── underscore.string.js
```

一目了然，我们可以归纳出下面几类主要文件：

- 入口：index.html
- 模块库：vendor/
- 程序入口：app/app.js
- 组件配置：app/components/
- 仪表板控制：app/controllers/
- 挂件页面：app/partials/
- 服务：app/services/
- 指令：app/directives/
- 图表：app/panels/

## 入口和模块依赖

---

这一部分是网页项目的基础。从 index.html 里就可以学到 angularjs 最基础的常用模板语法了。出现的指令有： `ng-repeat`，`ng-controller`，`ng-include`，`ng-view`，`ng-slow`，`ng-click`，`ng-href`，以及变量绑定的语法：`{{ dashboard.current.** }}`。

index.html 中，需要注意 js 的加载次序，先 `require.js`，然后再 `require.config.js`，最后 `app`。整个 kibana 项目都是通过 `require` 方式加载的。而具体的模块，和模块的依赖关系，则定义在 `require.config.js` 里。这些全部加载完成后，才是启动 `app` 模块，也就是项目本身的代码。

`require.config.js` 中，主要分成两部分配置，一个是 `paths`，一个是 `shim`。`paths` 用来指定依赖模块的导出名称和模块 js 文件的具体路径。而 `shim` 用来指定依赖模块之间的依赖关系。比方说：绘制图表的 js，kibana3 里用的是 `jquery.flot` 库。这个就首先依赖于 `jquery` 库。(通俗的说，就是原先普通的 HTML 写法里，要先加载 `jquery.js` 再加载 `jquery.flot.js`)

在整个 `paths` 中，需要单独提一下的是 `elasticjs:'./vendor/elasticjs/elastic-angular-client'`。这是串联 `elastic.js` 和 `angular.js` 的文件。这里面实际是定义了一个 `angular.module` 的 factory，名叫 `ejsResource`。后续我们在 kibana 3 里用到的跟 Elasticsearch 交互的所有方法，都在这个 `ejsResource` 里了。

`factory` 是 `angular` 的一个单例对象，创建之后会持续到你关闭浏览器。*Kibana 3* 就是通过这种方式来控制你所有的图表是从同一个 *Elasticsearch* 获取的数据

`app.js` 中，定义了整个应用的 routes，加载了 controller，directives 和 filters 里的全部内容。就是在这里，加载了主页面 `app/partials/dashboard.html`。当然，这个页面其实没啥看头，因为里面就是提供 pulldown 和 row 的 div，然后绑定到对应的 controller 上。

## controller 和 service

controller 里没太多可讲的。kibana 3 里， pulldown 其实跟 row 差别不大，看这简单的几行代码里，最关键的就是几个注入：

```
define(['angular','app','lodash'], function (angular, app, _) {
  'use strict';
  angular.module('kibana.controllers').controller('RowCtrl', function($scope, $rootScope, $timeout,ejsResource, querySrv) {
    var _d = {
      title: "Row",
      height: "150px",
      collapse: false,
      collapsable: true,
      editable: true,
      panels: [],
      notice: false
    };
    _.defaults($scope.row,_d);

    $scope.init = function() {
      $scope.querySrv = querySrv;
      $scope.reset_panel();
    };
    $scope.init();
  });
});
```

这里面，注入了 `$scope`，`ejsResource` 和 `querySrv`。`$scope` 是控制器作用域内的模型数据对象，这是 angular 提供的一个特殊变量。`ejsResource` 是一个 factory，前面已经讲过。`querySrv` 是一个 service，下面说一下。

service 跟 factory 的概念非常类似，一般来说，可能 factory 倾向用来共享一个类，而 service 用来共享一组函数功能。

kibana 3 里，比较有用和常用的 services 包括：

## dashboard

dashboard.js 里提供了关于 Kibana 3 仪表板的读写操作。其中主要的几个是提供了三种读取仪表板布局纲要的方式，也就是读取文件，读取存在 `.kibana-int` 索引里的数据，读取 js 脚本。下面是读取 js 脚本的相关函数：

```
this.script_load = function(file) {
  return $http({
    url: "app/dashboards/"+file.replace(/\.(?!js)/,"/"),
    method: "GET",
    transformResponse: function(response) {
      /*jshint -W054 */
      var _f = new Function('ARGS','kbn','_', 'moment', 'window', 'document', 'angular', 'require', 'define', '$', 'jQuery'
        return _f($routeParams,kbn,_,moment);
    }
  }).then(function(result) {
    if(!result) {
      return false;
    }
    self.dash_load(dash_defaults(result.data));
    return true;
  },function() {
    alertSrv.set('Error',
      "Could not load <i>scripts/" + file + "</i>. Please make sure it exists and returns a valid dashboard" ,
      'error');
    return false;
  });
};
```

可以看到，最关键的就是那个 `new Function`。知道这步传了哪些函数进去，也就知道你的 js 脚本里都可以调用哪些内容了~

最后调用的 `dash_load` 方法也需要提一下。这个方法的最后，有几行这样的代码：

```
self.availablePanels = _.difference(config.panel_names,  
_.pluck(_.union(self.current.nav, self.current.pulldowns), 'type'));  
  
self.availablePanels = _.difference(self.availablePanels, config.hidden_panels);
```

从最外层的 `config.js` 里读取了 `panel_names` 数组，然后取出了 `nav` 和 `pulldown` 用过的 panel，剩下就是我们能在 `row` 里添加的 panel 类型了。

## querySrv

`querySrv.js` 里定义了跟 `query` 框相关的函数和属性。主要有几个值得注意的。

- 一个是 `color` 列表；
- 一个是 `queryTypes`，尤其是里么的 `topN`，可以看到 `topN` 方式其实就是先请求了一次 `termsFacet`，然后把结果 map 成一组普通的 `query`。
- 一个是 `ids` 和 `idsByMode`。之后图表的绑定具体 `query` 的时候，就是通过这个函数来选择的。

## filterSrv

`filterSrv.js` 跟 `querySrv` 相似。特殊的是两个函数。

- 一个是 `toEjsObjs`。根据不同的 `filter` 类型调用不同的 `ejs` 方法。
- 一个是 `timeRange`。因为在 `histogram` panel 上拖拽，会生成好多个 `range` 过滤器，都是时间。这个方法会选择最后一个类型为 `time` 的 `filter`，作为实际要用的 `filter`。这样保证请求 ES 的是最后一次拖拽选定的时间段。

## fields

`fields.js` 里最重要的作用就是通过 `mapping` 接口获取索引的字段列表，存在 `fields.list` 里。这个数组后来在每个 panel 的编辑页里，都以 `bs-typeahead="fields.list"` 的形式作为文本输入时的自动补全提示。在 `table` panel 里，则是左侧栏的显示来源。

## esVersion

`esVersion.js` 里提供了对 ES 版本号的对比函数。之所以专门提供这么个 service，一来是因为不同版本的 ES 接口有变化，比如我自己开发的 `percentile` panel 里，就用 `esVersion` 判断了两次版本。因为 `percentile` 接口是 1.0 版之后才有，而从 1.3 版以后返回数据的结构又发生了一次变动。二来 ES 的版本号格式比较复杂，又有点又有字母。

# panel 相关指令

## 添加 panel

前面在讲 `app/services/dashboard.js` 的时候，已经说到能添加的 panel 列表是怎么获取的。那么 panel 是怎么加上的呢？

同样是之前讲过的 `app/partials/dashboard.html` 里，加载了 `partials/roweditor.html` 页面。这里有一段：

```
<form class="form-inline">
  <select class="input-medium" ng-model="panel.type" ng-options="panelType for panelType in dashboard.availablePanels"
    <small ng-show="rowSpan(row) > 11">
      Note: This row is full, new panels will wrap to a new line. You should add another row.
    </small>
  </form>

  <div ng-show="!(_.isUndefined(panel.type))">
    <div add-panel="{{panel.type}}></div>
  </div>
```

这个 `add-panel` 指令，是有 `app/directives/addPanel.js` 提供的。方法如下：

```
$scope.$watch('panel.type', function() {
  var _type = $scope.panel.type;
  $scope.reset_panel(_type);
  if(!_.isUndefined($scope.panel.type)) {
    $scope.panel.loadingEditor = true;
    $scope.require(['panels/'+$scope.panel.type.replace('.','/') +'/module'], function () {
      var template = '<div ng-controller="'+$scope.panel.type+'>' ng-include="\app/partials/paneladd.html\'';
      elem.html($compile(angular.element(template))($scope));
      $scope.panel.loadingEditor = false;
    });
  }
});
```

可以看到，其实就是 `require` 了对应的 `panels/xxx/module.js`，然后动态生成一个 `div`，绑定到对应的 controller 上。

## 展示 panel

还是在 `app/partials/dashboard.html` 里，用到了另一个指令 `kibana-panel`：

```
<div
  ng-repeat="(name, panel) in row.panels|filter:isPanel"
  ng-cloak ng-hide="panel.hide"
  kibana-panel type='panel.type' resizable
  class="panel nospace" ng-class="{'dragInProgress':dashboard.panelDragging}"
  style="position:relative" ng-style="{'width':!panel.span?'100%':((panel.span/1.2)*10)+'%'}"
  data-drop="true" ng-model="row.panels" data-jqyoui-options
  jqyoui-droppable="{index:$index,mutate:false,onDrop:'panelMoveDrop',onOver:'panelMoveOver(true)',onOut:'p
</div>
```

当然，这里面还有 `resizable` 指令也是自己实现的，不过一般我们用不着关心这个的代码实现。

下面看 `app/directives/kibanaPanel.js` 里的实现。

这个里面大多数逻辑跟 `addPanel.js` 是一样的，都是为了实现一个指令嘛。对于我们来说，关注点在前面那一大段 HTML 字符串，也就是变量 `panelHeader`。这个就是我们看到的实际效果中，kibana 3 每个 panel 顶部那个小图标工具栏。仔细阅读一下，可以发现除了每个 panel 都一致的那些 `span` 以外，还有一段是：

```
'<span ng-repeat="task in panelMeta.modals" class="row-button extra" ng-show="task.show">' +  
'<span bs-modal="task.partial" class="pointer"><i ' +  
'bs-tooltip="task.description" ng-class="task.icon" class="pointer"></i></span>' +  
'</span>'
```

也就是说，每个 panel 可以在自己的 panelMeta.modals 数组里，定义不同的小图标，弹出不同的对话浮层。我个人给 table panel 二次开发加入的 exportAsCsv 功能，图标就是在这里加入的。

# panel 内部实现

终于说到最后了。大家进入到 `app/panels/` 下，每个目录都是一种 panel。原因前一节已经分析过了，因为 `addPanel.js` 里就是直接这样拼接的。入口都是固定的：`module.js`。

下面以 stats panel 为例。(因为我最开始就是抄的 stats 做的 percentile，只有表格没有图形，最简单)

每个目录下都会有至少一下三个文件：

## module.js

`module.js` 就是一个 controller。跟前面讲过的 controller 写法其实是一致的。在 `$scope` 对象上，有几个属性是 panel 实现时一般都会有的：

- `$scope.panelMeta`：这个前面说到过，其中的 `modals` 用来定义 `panelHeader`。
- `$scope.panel`：用来定义 panel 的属性。一般实现上，会有一个 `default` 值预定义好。你会发现这个 `$scope.panel` 其实就是仪表纲要里面说的每个 panel 的可设置值！

然后一般 `$scope.init()` 都是这样的：

```
$scope.init = function () {
  $scope.ready = false;
  $scope.$on('refresh', function () {
    $scope.get_data();
  });
  $scope.get_data();
};
```

也就是每次有刷新操作，就执行 `get_data()` 方法。这个方法就是获取 ES 数据，然后渲染效果的入口。

```
$scope.get_data = function () {
  if(dashboard.indices.length === 0) {
    return;
  }

  $scope.panelMeta.loading = true;

  var request,
    results,
    boolQuery,
    queries;

  request = $scope.ejs.Request();

  $scope.panel.queries.ids = querySrv.idsByMode($scope.panel.queries);
  queries = querySrv.getQueryObjs($scope.panel.queries.ids);

  boolQuery = $scope.ejs.BoolQuery();
  _.each(queries, function(q) {
    boolQuery = boolQuery.should(querySrv.toEjsObj(q));
  });

  request = request
    .facet($scope.ejs.StatisticalFacet('stats'))
    .field($scope.panel.field)
    .facetFilter($scope.ejs.QueryFilter(
      $scope.ejs.FilteredQuery(
        boolQuery,
        filterSrv.getBoolFilter(filterSrv.ids())
      )
    )).size(0);

  _.each(queries, function (q) {
    var alias = q.alias || q.query;
    var query = $scope.ejs.BoolQuery();
    query.should(querySrv.toEjsObj(q));
    request.facet($scope.ejs.StatisticalFacet('stats_'+alias)
      .field($scope.panel.field)
```

```

    .facetFilter($scope.ejs.QueryFilter(
      $scope.ejs.FilteredQuery(
        query,
        filterSrv.getBoolFilter(filterSrv.ids())
      )
    );
  });
};

$scope.inspector = request.toJSON();

results = $scope.ejs.doSearch(dashboard.indices, request);

results.then(function(results) {
  $scope.panelMeta.loading = false;
  var value = results.facets.stats[$scope.panel.mode];

  var rows = queries.map(function (q) {
    var alias = q.alias || q.query;
    var obj = _.clone(q);
    obj.label = alias;
    obj.Label = alias.toLowerCase(); //sort field
    obj.value = results.facets['stats_'+alias];
    obj.Value = results.facets['stats_'+alias]; //sort field
    return obj;
  });

  $scope.data = {
    value: value,
    rows: rows
  };

  $scope.$emit('render');
});
});

```

stats panel 的这段函数几乎就跟基础示例一样了。

1. 生成 Request 对象。
2. 获取关联的 query 对象。
3. 获取当前页的 filter 对象。
4. 调用选定的 facets 方法，传入参数。
5. 如果有多个 query，逐一构建 facets。
6. request 完成。生成一个 JSON 内容供 inspector 查看。
7. 发送请求，等待异步回调。
8. 回调处理数据成绑定在模板上的 \$scope.data。
9. 渲染页面。

注：stats/module.js 后面还有一个 filter，terms/module.js 后面还有一个 directive，这些都是为了实际页面效果加的功能，跟 kibana 本身的 filter，directive 本质上是一样的。就不单独讲述了。

## module.html

module.html 就是 panel 的具体页面内容。没有太多可说的。大概框架是：

```

<div ng-controller='stats' ng-init="init()">
  <table ng-style="panel.style" class="table table-striped table-condensed" ng-show="panel.chart == 'table'">
    <thead>
      <th>Term</th> <th>{{ panel.tmode == 'terms_stats' ? panel.tstat : 'Count' }}</th> <th>Action</th>
    </thead>
    <tr ng-repeat="term in data" ng-show="showMeta(term)">
      <td class="terms-legend-term">{{term.label}}</td>
      <td>{{term.data[0][1]}}</td>
    </tr>
  </table>
</div>

```

主要就是绑定要 controller 和 init 函数。对于示例的 stats，里面的 data 就是 module.js 最后生成的 \$scope.data。

## editor.html

editor.html 是 panel 参数的编辑页面主要内容，参数编辑还有一些共同的标签页，是在 kibana 的 `app/partials/` 里，就不讲了。

editor.html 里，主要就是提供对 `$scope.panel` 里那些参数的修改保存操作。当然实际上并不是所有参数都暴露出来了。这也是 kibana 3 用户指南里，官方说采用仪表板纲要，比通过页面修改更灵活细腻的原因。

editor.html 里需要注意的是，为了每次变更都能实时生效，所有的输入框都注册到了刷新事件。所以一般是这样子：

```
<select ng-change="set_refresh(true)" class="input-small" ng-model="panel.format" ng-options="f for f in ['number
```

这个 `set_refresh` 函数是在 `module.js` 里定义的：

```
$scope.set_refresh = function (state) {  
    $scope.refresh = state;  
};
```

## 总结

---

kibana 3 源码的主体分析，就是这样了。怎么样，看完以后，大家有没有信心也做些二次开发，甚至跟 grafana 一样，替换掉 `esResource`，换上一个你自己的后端数据源呢？

# README

Kibana 是为 Elasticsearch 设计的开源分析和可视化平台。你可以使用 Kibana 来搜索，查看存储在 Elasticsearch 索引中的数据并与之交互。你可以很容易实现高级的数据分析和可视化，以图标的形式展现出来。

Kibana 让海量数据变得更容易理解。简单的基于浏览器的界面让你可以快速创建并分享动态的仪表板，用以实时修改 Elasticsearch 请求。

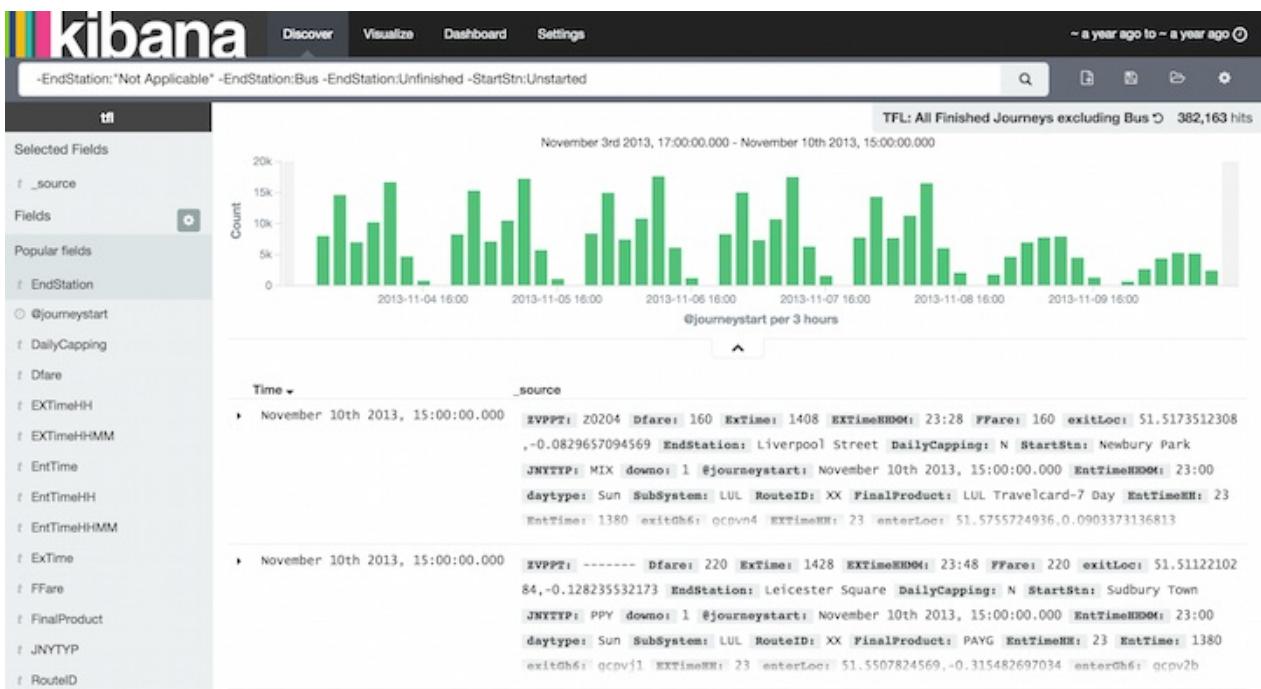
安装 Kibana 非常简单。你可以在几分钟内安装好 Kibana 然后开始探索你的 Elasticsearch 索引——不需要写代码，不需要额外的架构。

本指南讲述的是如何使用 Kibana 4。想了解 Kibana 4 里有什么新特性，请阅读 [What's New in Kibana 4](#)。想了解 Kibana 3 的内容，请阅读 [Kibana 3 用户指南](#)。

## 数据发现和可视化

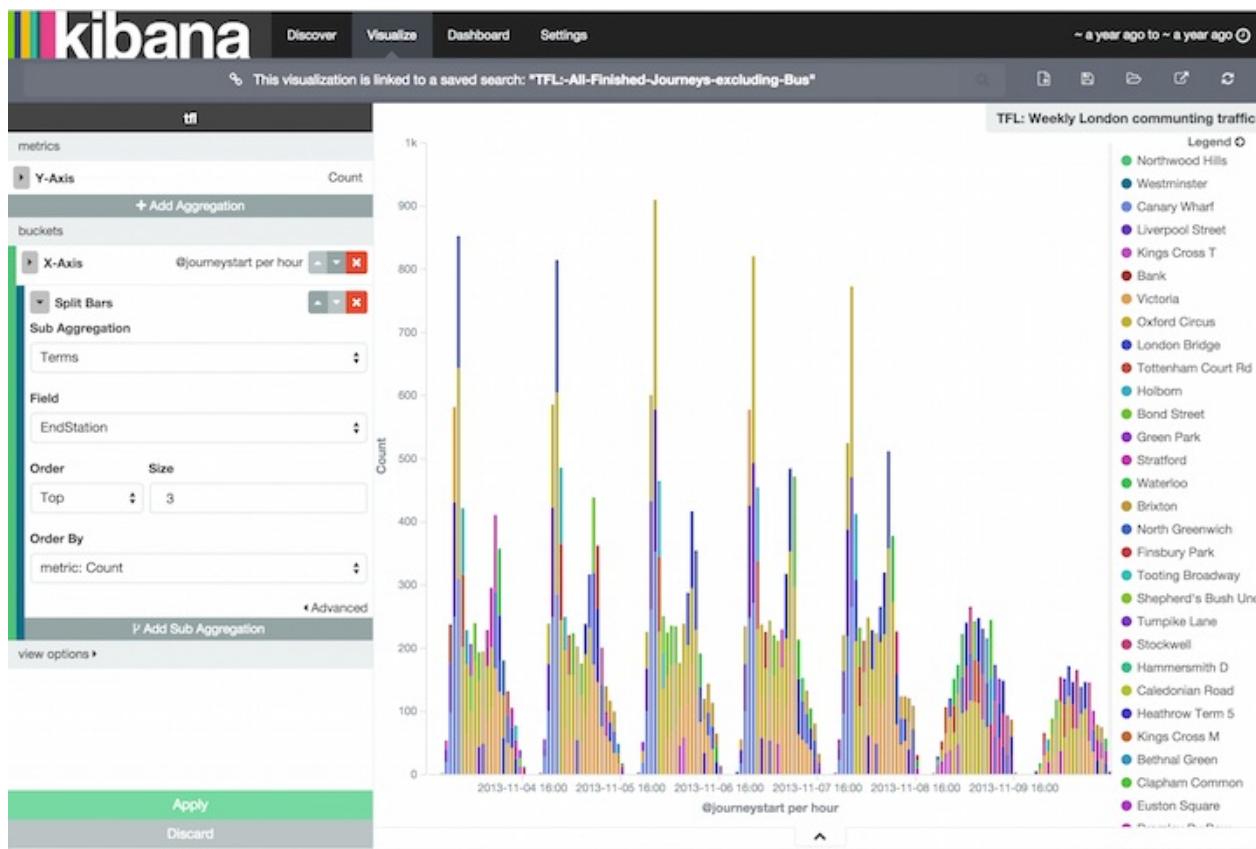
让我们看看你可能要怎么用 Kibana 来探索和展示数据。我们会从伦敦交通局的交通运输卡的一周使用情况里导入一些数据。

在 Kibana 的 Discover 页，我们可以提交搜索请求，过滤结果，然后检查返回的文档里的数据。比如，我们可以通过排除公交出行，获取地铁出行的情况。

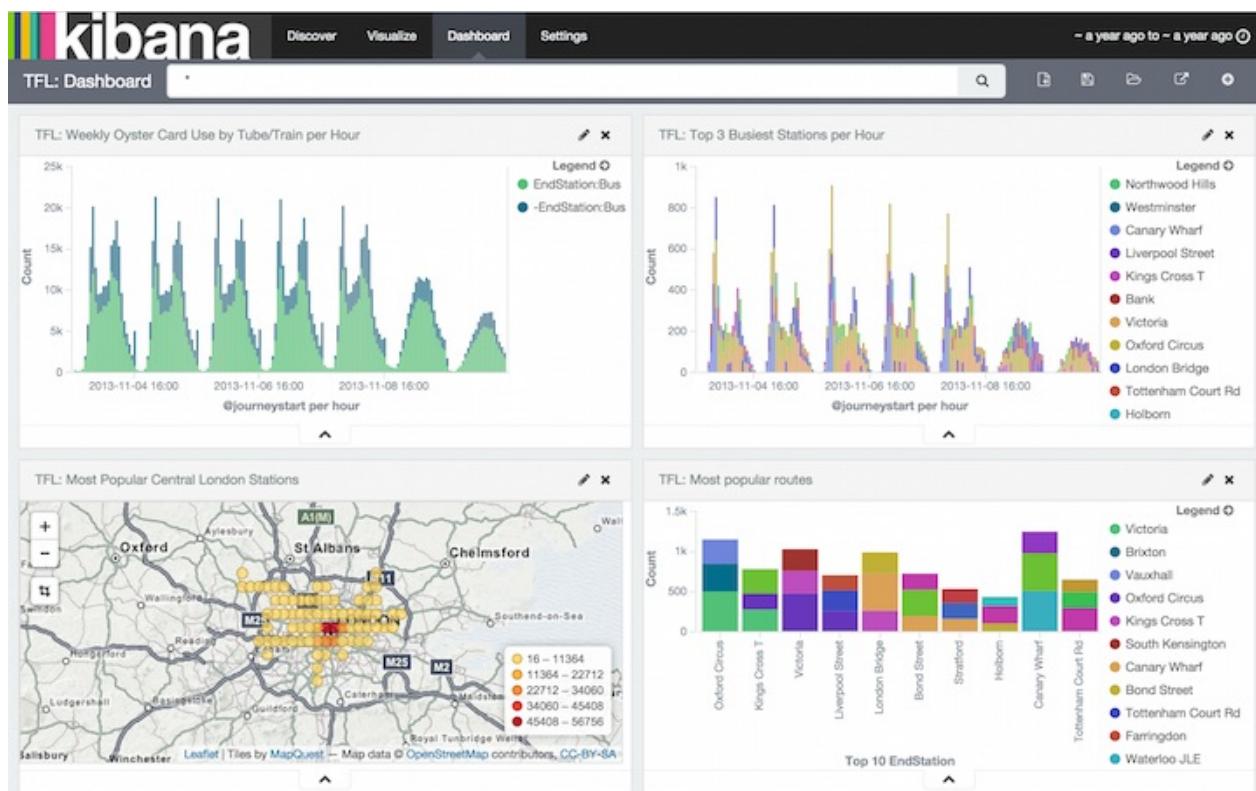


现在，我们可以看到早晚上下班高峰期的直方图。默认情况下，Discover 页会显示匹配搜索条件的前 500 个文档。你可以修改时间过滤器，拖拽直方图下钻数据，查看部分文档的细节。Discover 页上如何探索数据，详细说明见 [Discover](#)。

你可以在 Visualization 页为你的搜索结构构造可视化。每个可视化都是跟一个搜索关联着的。比如，我们可以基于前面那个搜索创建一个每天伦敦地铁交通流量的直方图。Y 轴显示交通流量。X 轴显示时间。而添加一个子聚合，我们还可以看到每小时排名前三的地铁站。



你可以保存并分析可视化结果，然后合并到仪表板上以便对比分析。比如说，我们可以创建一个展示多个伦敦交通数据的仪表板：



更多关于创建和分享可视化和仪表板的内容，请阅读 [Visualize](#) 和 [Dashboard](#) 章节。

你可以在几分钟内安装好 Kibana 然后开始探索你的 Elasticsearch 索引。你要的就是：

- Elasticsearch 1.4.4 或者更新的版本
- 一个现代浏览器 - [支持的浏览器列表](#).
- 有关你的 Elasticsearch 集群的信息：
  - 你想要连接 Elasticsearch 实例的 URL
  - 你想搜索哪些 Elasticsearch 紴引

如果你的 Elasticsearch 是被 [Shield](#) 保护着的，阅读 [Shield with Kibana 4](#) 学习额外的安装说明。

## 安装并启动 kibana

要安装启动 Kibana:

1. 下载对应平台的 [Kibana 4 二进制包](#)
2. 解压 .zip 或 tar.gz 压缩文件
3. 在安装目录里运行: bin/kibana (Linux/MacOSX) 或 bin\kibana.bat (Windows)

完毕！Kibana 现在运行在 5601 端口了。

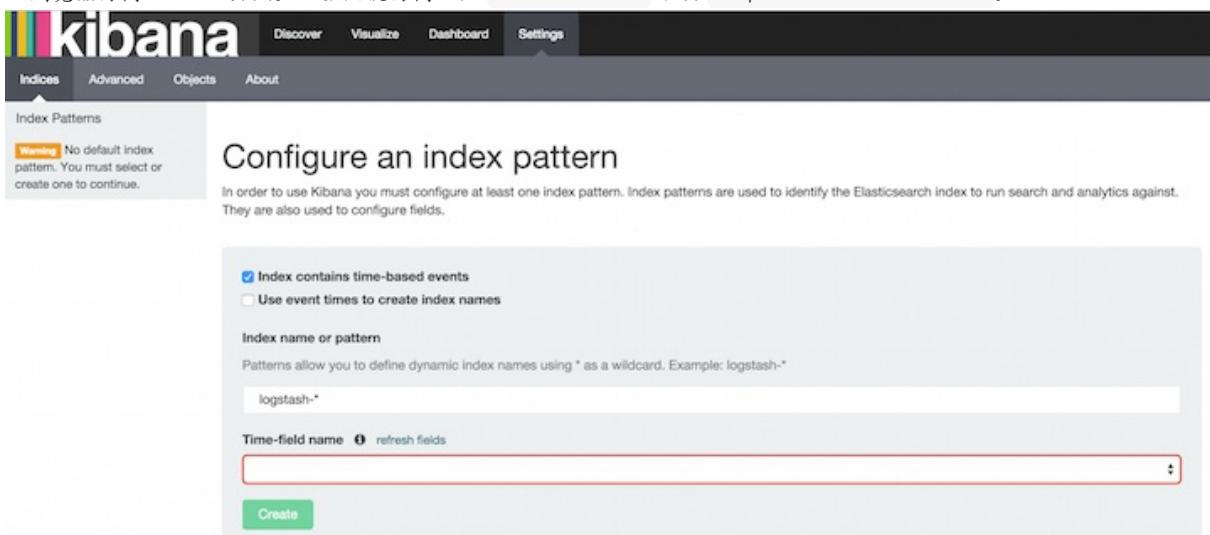
## 让 kibana 连接到 elasticsearch

在开始用 Kibana 之前，你需要告诉它你打算探索哪个 Elasticsearch 紴引。第一次访问 Kibana 的时候，你会被要求定义一个 *index pattern* 用来匹配一个或者多个索引名。好了。这就是你需要做的全部工作。以后你还可以随时从 [Settings tab](#) 页面添加更多的 index pattern。

默认情况下，Kibana 会连接运行在 localhost 的 Elasticsearch。要连接其他 Elasticsearch 实例，修改 kibana.yml 里的 Elasticsearch URL，然后重启 Kibana。如何在生产环境下使用 Kibana，阅读 [Using Kibana in a Production Environment](#)。

要从 Kibana 访问的 Elasticsearch 紹引的配置方法：

1. 从浏览器访问 Kibana 界面。也就是说访问比如 localhost:5601 或者 http://YOURDOMAIN.com:5601。



2. 制定一个可以匹配一个或者多个 Elasticsearch 紹引的 index pattern。默认情况下，Kibana 认为你要访问的是通过 Logstash 导入 Elasticsearch 的数据。这时候你可以用默认的 logstash-\* 作为你的 index pattern。通配符(\*) 匹配紹引名中零到多个字符。如果你的 Elasticsearch 紹引有其他命名约定，输入合适的 pattern。pattern 也开始是最简单的单个紹引的名字。
3. 选择一个包含了时间戳的紹引字段，可以用来做基于时间的处理。Kibana 会读取紹引的映射，然后列出所有包含了时间戳的字段(译者注：实际是字段类型为 date 的字段，而不是“看起来像时间戳”的字段)。如果你的紹引没有基于时间的数据，关闭 Index contains time-based events 参数。

4. 如果一个新索引是定期生成，而且索引名中带有时间戳，选择 `Use event times to create index names` 选项，然后再选择 `Index pattern interval`。这可以提高搜索性能，Kibana 会至搜索你指定的时间范围内的索引。在你用 Logstash 输出数据给 Elasticsearch 的情况下尤其有效。
5. 点击 `Create` 添加 index pattern。第一个被添加的 pattern 会自动被设置为默认值。如果你有多个 index pattern 的时候，你可以在 `Settings > Indices` 里设置具体哪个是默认值。

好了。Kibana 现在连接上你的 Elasticsearch 数据了。Kibana 会显示匹配上的索引里的字段名的只读列表。

## 开始探索你的数据！

---

你可以开始下钻你的数据了：

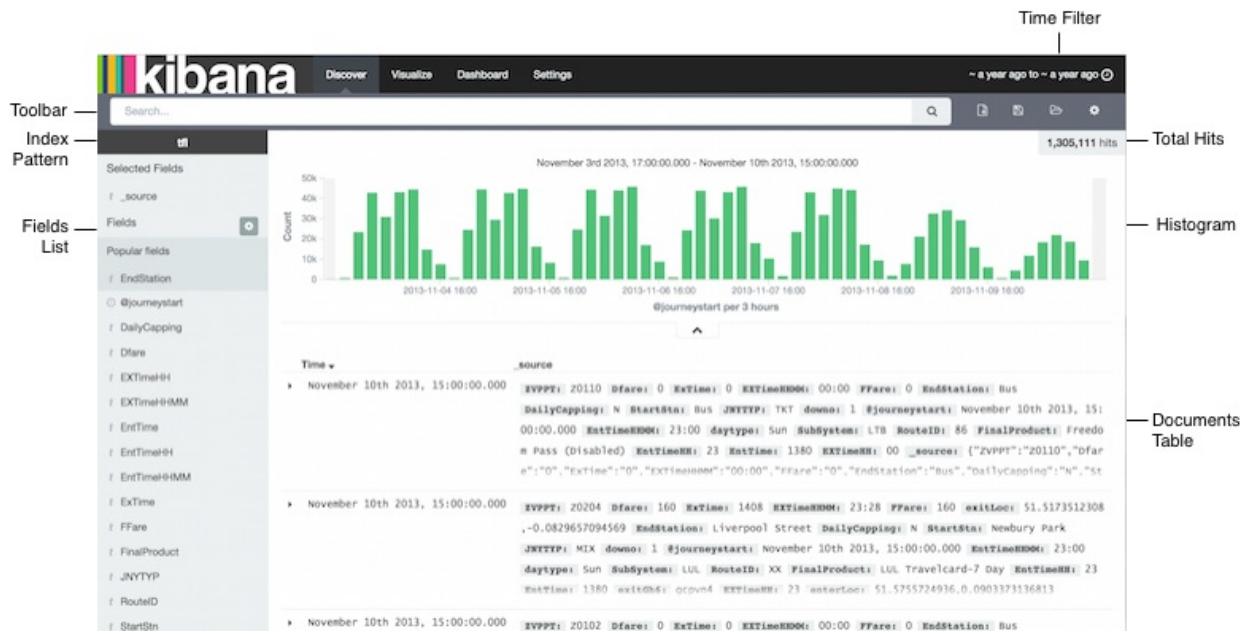
- 在 [Discover](#) 页搜索和浏览你的数据。
- 在 [Visualize](#) 页转换数据成图表。
- 在 [Dashboard](#) 页创建定制自己的仪表板。

Kibana 是让你从 5601 端口访问的网页应用。你需要做的只是打开浏览器，然后输入你运行 Kibana 的机器地址然后加上端口号。比如说：`localhost:5601` 或者 `http://YOURDOMAIN.com:5601`。

当你访问 Kibana 的时候，默认会加载 Discover 页以及默认的索引模式。时间选择器默认为最近 15 分钟。而搜索条件是全部匹配(\*)。

如果你没看到任何文档，尝试放宽时间选择器范围。如果还没有，可能你确实没往 Elasticsearch 里写进数据。

你可以在 Discover 页交互式探索你的数据。你可以访问到匹配得上你选择的索引模式的每个索引的每条记录。你可以提交搜索请求，过滤搜索结果，然后查看文档数据。你还可以看到匹配搜索请求的文档总数，获取字段值的统计情况。如果索引模式配置了时间字段，文档的时序分布情况会在页面顶部以柱状图的形式展示出来。



## 设置时间过滤器

时间过滤器(Time Filter)限制搜索结果在一个特定的时间周期内。如果你的索引包含的是时序语句，而且你为所选的索引模式配置了时间字段，那么就可以设置时间过滤器。

默认的时间过滤器设置为最近 15 分钟。你可以用页面顶部的时间选择器(Time Picker)来修改时间过滤器，或者选择一个特定的时间间隔，或者直方图的时间范围。

要用时间选择器来修改时间过滤器：

1. 点击菜单栏右上角显示的 Time Filter 打开时间选择器。
2. 快速过滤，直接选择一个短链接即可。
3. 要指定相对时间过滤，点击 Relative 然后输入一个相对的开始时间。可以是任意数字的秒、分、小时、天、月甚至年之前。
4. 要指定绝对时间过滤，点击 Absolute 然后在 From 框内输入开始日期，To 框内输入结束日期。
5. 点击时间选择器底部的箭头隐藏选择器。

要从住房图上设置时间过滤器，有以下几种方式：

- 想要放大那个时间间隔，点击对应的柱体。
- 单击并拖拽一个时间区域。注意需要等到光标变成加号，才意味着这是一个有效的起始点。

你可以用浏览器的后退键来回退你的操作。

## 搜索数据

在 Discover 页提交一个搜索，你就可以搜索匹配当前索引模式的索引数据了。你可以直接输入简单的请求字符串，也就是用 Lucene query syntax，也可以用完整的基于 JSON 的 Elasticsearch Query DSL。

当你提交搜索的时候，直方图，文档表格，字段列表，都会自动反映成搜索的结果。hits(匹配的文档)总数会在直方图的右上角显示。文档表格显示前 500 个匹配文档。默认的，文档倒序排列，最新的文档最先显示。你可以通过点击时间列的头部来反转排序。事实上，所有建了索引的字段，都可以用来排序。更多细节，请阅读 [Sorting the Documents Table](#)。

要搜索你的数据：

## 1. 在搜索框内输入请求字符串：

- 简单的文本搜索，直接输入文本字符串。比如，如果你在搜索网站服务器日志，你可以输入 `safari` 来搜索各字段中的 `safari` 单词。
- 要搜索特定字段中的值，则在值前加上字段名。比如，你可以输入 `status:200` 来限制搜索结果都是在 `status` 字段里有 `200` 内容。
- 要搜索一个值的范围，你可以用范围查询语法，`[START_VALUE TO END_VALUE]`。比如，要查找 `4xx` 的状态码，你可以输入 `status:[400 TO 499]`。
- 要指定更复杂的搜索标准，你可以用布尔操作符 `AND`，`OR`，和 `NOT`。比如，要查找 `4xx` 的状态码，还是 `php` 或 `html` 结尾的数据，你可以输入 `status:[400 TO 499] AND (extension:php OR extension:html)`。

这些例子都用了 Lucene query syntax。你也可以提交 Elasticsearch Query DSL 式的请求。更多示例，请阅读 Elasticsearch 文档中的 [query string syntax](#)。

1. 点击回车键，或者点击 `Search` 按钮提交你的搜索请求。

## 开始一个新的搜索

要清除当前搜索或开始一个新搜索，点击 Discover 工具栏的 `New Search` 按钮。



## 保存搜索

你可以在 Discover 页加载已保存的搜索，也可以用作 [visualizations](#) 的基础。保存一个搜索，意味着同时保存下了搜索请求字符串和当前选择的索引模式。

要保存当前搜索：

1. 点击 Discover 工具栏的 `Save Search` 按钮 .
2. 输入一个名称，点击 `Save`。

## 加载一个已存搜索

要加载一个已保存的搜索：

1. 点击 Discover 工具栏的 `Load Search` 按钮 .
2. 选择你要加载的搜索。

如果已保存的搜索关联到跟你当前选择的索引模式不一样的其他索引上，加载这个搜索也会切换当前的已选索引模式。

## 改变你搜索的索引

当你提交一个搜索请求，匹配当前的已选索引模式的索引都会被搜索。当前模式模式会显示在搜索栏下方。要改变搜索的索引，需要选择另外的模式模式。

要选择另外的索引模式：

1. 点击 Discover 工具栏的 `Settings` 按钮 .
2. 从索引模式列表中选取你打算采用的模式。

关于索引模式的更多细节，请阅读 [Creating an Index Pattern](#)。

## 自动刷新页面

亦可以配置一个刷新间隔来自动刷新 Discover 页面的最新索引数据。这回定期重新提交一次搜索请求。

设置刷新间隔后，会显示在菜单栏时间过滤器的左边。

要设置刷新间隔：



1. 点击菜单栏右上角的 Time Filter
2. 点击 Refresh Interval 标签。
3. 从列表中选择一个刷新间隔。

## 按字段过滤

你可以过滤搜索结果，只显示在某字段中包含了特定值的文档。也可以创建反向过滤器，排除掉包含特定字段值的文档。

你可以从字段列表或者文档表格里添加过滤器。当你添加好一个过滤器后，它会显示在搜索请求下方的过滤栏里。从过滤栏里你可以编辑或者关闭一个过滤器，转换过滤器(从正向改成反向，反之亦然)，切换过滤器开关，或者完全移除掉它。

要从字段列表添加过滤器：

1. 点击你想要过滤的字段名。会显示这个字段的前 5 名数据。每个数据的右侧，有两个小按钮——一个用来添加常规(正向)过滤器，一个用来添加反向过滤器。
2. 要添加正向过滤器，点击 Positive Filter 按钮 。这个会过滤掉在本字段不包含这个数据的文档。
3. 要添加反向过滤器，点击 Negative Filter 按钮 。这个会过滤掉在本字段包含这个数据的文档。

要从文档表格添加过滤器：

1. 点击表格第一列(通常都是时间)文档内容左侧的 Expand 按钮 展开文档表格中的文档。每个字段名的右侧，有两个小按钮——一个用来添加常规(正向)过滤器，一个用来添加反向过滤器。
2. 要添加正向过滤器，点击 Positive Filter 按钮 。这个会过滤掉在本字段不包含这个数据的文档。
3. 要添加反向过滤器，点击 Negative Filter 按钮 。这个会过滤掉在本字段包含这个数据的文档。

## 查看文档数据

当你提交一个搜索请求，最近的 500 个搜索结果会显示在文档表格里。你可以在 [Advanced Settings](#) 里通过 `discover:sampleSize` 属性配置表格里具体的文档数量。默认的，表格会显示当前选择的索引模式中定义的时间字段内容(转换成本地时区)以及 `_source` 文档。你可以从字段列表[添加字段到文档表格](#)。还可以用表格里包含的任意已建索引的字段来[排列列出的文档](#)。

要查看一个文档的字段数据：

1. 点击表格第一列(通常都是时间)文档内容左侧的 Expand 按钮 。Kibana 从 Elasticsearch 读取数据然后在表格中显示文档字段。这个表格每行是一个字段的名字、过滤器按钮和字段的值。
2. 要查看原始 JSON 文档(格式美化过的)，点击 JSON 标签。
3. 要在单独的页面上查看文档内容，点击链接。你可以添加书签或者分享这个链接，以直接访问这条特定文档。
4. 收回文档细节，点击 collapse 按钮 .

## 文档列表排序

你可以用任意已建索引的字段排序文档表格中的数据。如果当前索引模式配置了时间字段，默认会使用该字段倒序排列文档。

要改变排序方式：

- 点击想要用来排序的字段名。能用来排序的字段在字段名右侧都有一个排序按钮。再次点击字段名，就会反向调整排序方式。

## 给文档表格添加字段列

By default, the Documents table shows the localized version of the time field specified in the selected index pattern and the document `_source`. You can add fields to the table from the Fields list. 默认的，文档表格会显示当前选择的索引模式中定义的时间字段内容(转换成本地时区)以及 `_source` 文档。你可以从字段列表添加字段到文档表格。

要添加字段列到文档表格：

- 移动鼠标到字段列表的字段上，点击它的 `add` 按钮 
- 重复操作直到你添加完所有你想移除的字段。

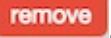
添加的字段会替换掉文档表格里的 `_source` 列。同时还会显示在字段列表顶部的 `Selected Fields` 区域里。

要重排表格中的字段列，移动鼠标到你要移动的列顶部，点击移动过按钮。

Time ^	index	@message ^ <
February 14th 2015, 10:36:51.075	logstash-2015.02.14	1 [Move column to the left] 2015-02-14T18:36:51.075Z] "GET /canhaz/yelena-kondakova.gif HTTP/1.1" 200 546 "-" "Mozilla/5.0 (X11; Linux i686) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.50 Safari/534.24"

## 从文档表格删除字段列

要从文档表格删除字段列：

- 移动鼠标到字段列表的 `Selected Fields` 区域里你想要移除的字段上，然后点击它的 `remove` 按钮 
- 重复操作直到你移除完所有你想移除的字段。

## 查看字段数据统计

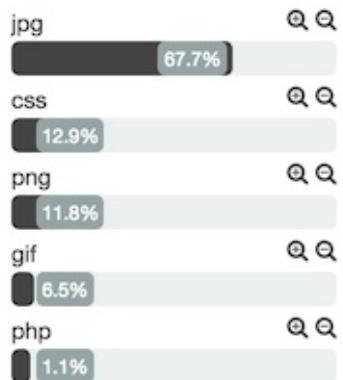
从字段列表，你可以看到文档表格里有多少数据包含了这个字段，排名前 5 的值是什么，以及包含各个值的文档的占比。

要查看字段数据统计：

- 点击字段列表里一个字段的名字。这个字段可以在字段列表的任意位置——已选字段(Selected Fields)，常用字段(Popular Fields)，或其他字段。

extension

Quick Count ( 93 /93 records )



要基于这个字段创建可视化，点击字段统计下方的 Visualize 按钮。

你可以用 **Visualize** 页来设计可视化。你可以保存可视化，以后再用，或者合并到 **dashboard** 里。一个可视化可以基于以下几种数据源类型：

- 一个新的交互式搜索
- 一个已保存的搜索
- 一个已保存的可视化

可视化是基于 Elasticsearch 1 引入的**聚合(aggregation)** 特性。

## 创建一个新可视化

要开始一个 **New Visualization** 向导，点击页面左上角的 **Visualize** 标签。如果你已经在创建一个可视化了。你可以在搜索栏的右侧工具栏里点击 **New Visualization** 按钮  向导会引导你继续以下几步：

### 第 1 步：选择可视化类型

**New Visualization** 向导起始页如下：

**Create a new visualization**

**Step 1**

Icon	Chart Type	Description
	Area chart	Great for stacked timelines in which the total of all series is more important than comparing any two or more series. Less useful for assessing the relative change of unrelated data points as changes in a series lower down the stack will have a difficult to gauge effect on the series above it.
	Data table	The data table provides a detailed breakdown, in tabular format, of the results of a composed aggregation. Tip: a data table is available from many other charts by clicking grey bar at the bottom of the chart.
	Line chart	Often the best chart for high density time series. Great for comparing one series to another. Be careful with sparse sets as the connection between points can be misleading.
	Markdown widget	Useful for displaying explanations or instructions for dashboards.
	Metric	One big number for all of your one big number needs. Perfect for show a count of hits, or the exact average a numeric field.
	Pie chart	Pie charts are ideal for displaying the parts of some whole. For example, sales percentages by department. Pro Tip: Pie charts are best used sparingly, and with no more than 7 slices per pie.
	Tile map	Your source for geographic maps. Requires an elasticsearch geo_point field. More specifically, a field that is mapped as type:geo_point with latitude and longitude coordinates.
	Vertical bar chart	The goto chart for oh-so-many needs. Great for time and non-time data. Stacked or grouped, exact numbers or percentages. If you are not sure which chart your need, you could do worse than to start here.

你也可以加载一个你之前创建好保存下来的可视化。已存可视化选择器包括一个文本框用来过滤可视化名称，以及一个指向**对象编辑器(Object Editor)** 的链接，可以通过 **Settings > Edit Saved Objects** 来管理已存的可视化。

如果你的新可视化是一个 **Markdown** 挂件，选择这个类型会带你到一个文本内容框，你可以在框内输入打算显示在挂件里的文本。其他的可视化类型，选择后都会转到数据源选择。

### 第 2 步：选择数据源

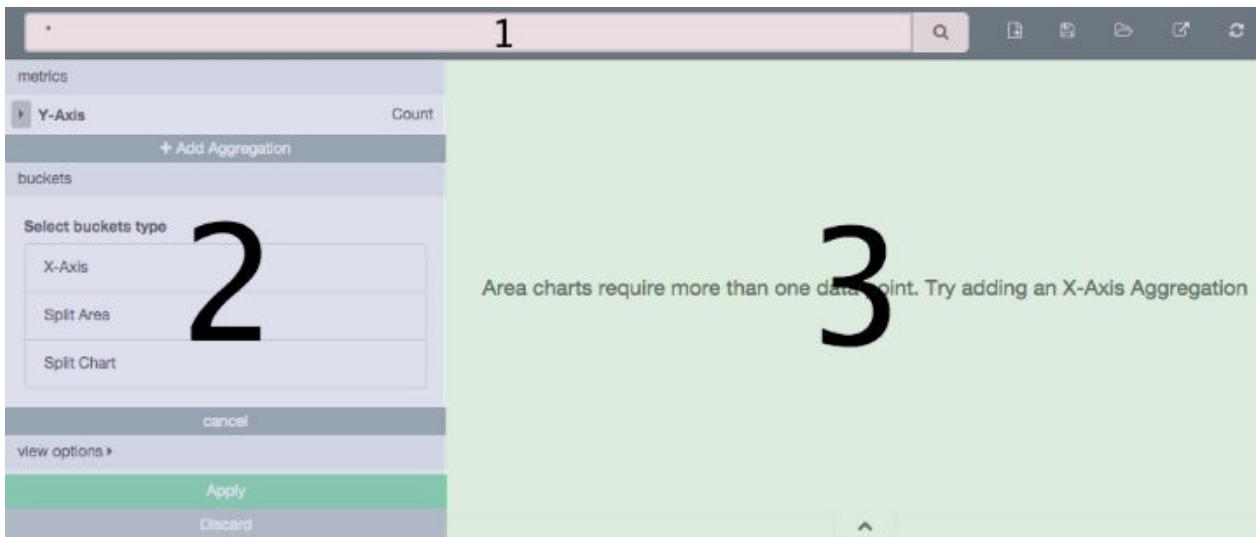
你可以选择新建或者读取一个已保存的搜索，作为你可视化的数据源。搜索是和一个或者一系列索引相关联的。如果你选择了在一个配置了多个索引的系统上开始你的新搜索，从可视化编辑器的下拉菜单里选择一个索引模式。

当你从一个已保存的搜索开始创建并保存好了可视化，这个搜索就绑定在这个可视化上。如果你修改了搜索，对应的可视化也会自动更新。

### 第 3 步：可视化编辑器

The visualization editor enables you to configure and edit visualizations. The visualization editor has the following main elements: 可视化编辑器用来配置编辑可视化。它有下面几个主要元素：

1. 工具栏(Toolbar)
2. 聚合构建器(Aggregation Builder)
3. 预览画布(Preview Canvas)



## 工具栏

工具栏上有一个用户交互式数据搜索的搜索框，用来保存和加载可视化。因为可视化是基于保存好的搜索，搜索栏会变成灰色。要编辑搜索，双击搜索框，用编辑后的版本替换已保存搜索。

搜索框右侧的工具栏有一系列按钮，用于创建新可视化，保存当前可视化，加载一个已有可视化，分享或内嵌可视化，和刷新当前可视化的数据。

## 聚合构建器

用页面左侧的聚合构建器配置你的可视化要用的 `metric` 和 `bucket` 聚合。桶(Buckets)的效果类似于 SQL `GROUP BY` 语句。想更详细的了解聚合，阅读 [Elasticsearch aggregations reference](#)。

在条带图或者折线图可视化里，用 `metrics` 做 Y 轴，然后 `buckets` 做 X 轴，条带颜色，以及行/列的区分。在饼图里，`metrics` 用做分片的大小，`buckets` 做分片的数量。

为你的可视化 Y 轴选一个 `metric` 聚合，包括 `count`, `average`, `sum`, `min`, `max`, or `cardinality` (unique count). 为你的可视化 X 轴，条带颜色，以及行/列的区分选一个 `bucket` 聚合，常见的有 `date histogram`, `range`, `terms`, `filters`, 和 `significant terms`。

你可以设置 `buckets` 执行的顺序。在 Elasticsearch 里，第一个聚合决定了后续聚合的数据集。下面例子演示一个网页访问量前五名的文件后缀名统计的时间条带图。

要看所有相同后缀名的，设置顺序如下：

1. `Color` : 后缀名的 Terms 聚合
2. `X-Axis` : `@timestamp` 的时间条带图

Elasticsearch 收集记录，算出前 5 名后缀名，然后为每个后缀名创建一个时间条带图。

要看每个小时的前 5 名后缀名情况，设置顺序如下：

1. `X-Axis` : `@timestamp` 的时间条带图(1 小时间隔)
2. `Color` : 后缀名的 Terms 聚合

这次，Elasticsearch 会从所有记录里创建一个时间条带图，然后在每个桶内，分组(本例中就是一个小时的间隔)计算出前 5 名的后缀名。

记住，每个后续的桶，都是从前一个的桶里分割数据。

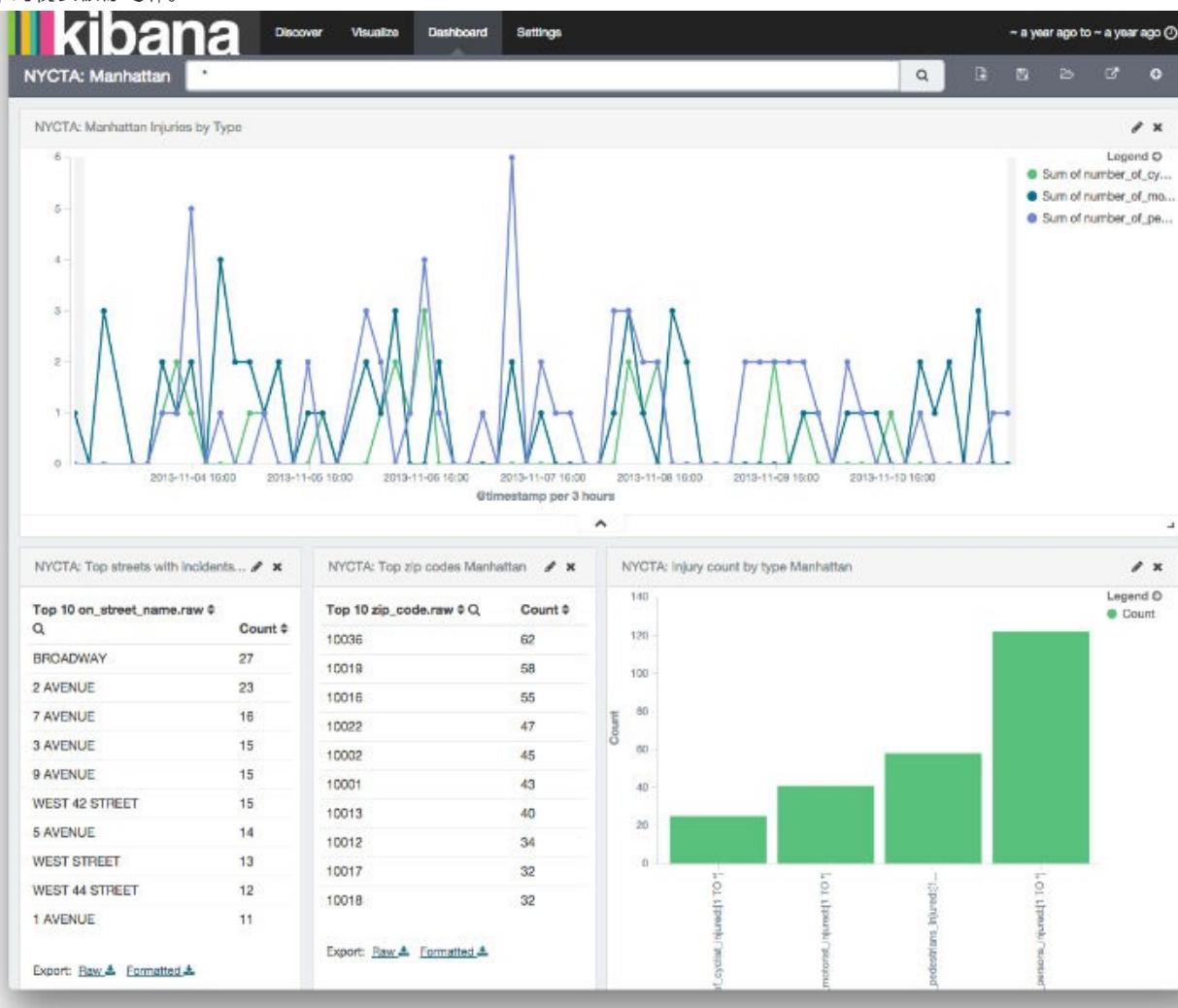
要在预览画布(*preview canvas*)上渲染可视化，点击聚合构建器底部的 **Apply** 按钮。

## 预览画布(canvas)

预览 canvas 上显示你定义在聚合构建器里的可视化的预览效果。要刷新可视化预览，点击工具栏里的 **Refresh** 按钮 。

一个 Kibana dashboard 能让你自由排列一组已保存的可视化。然后你可以保存这个仪表板，用来分享或者重载。

简单的仪表板像这样。

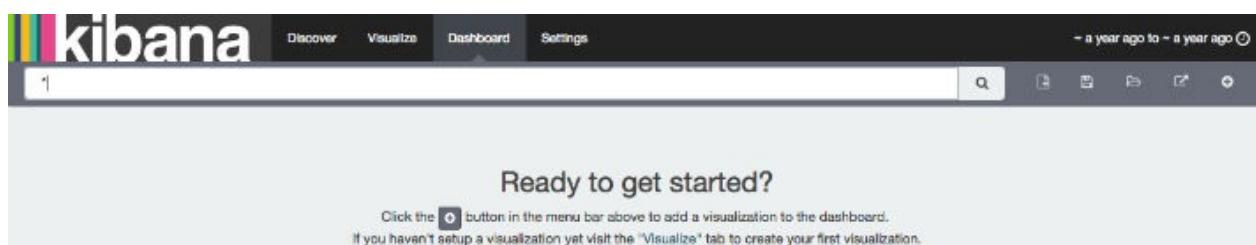


## 开始

要用仪表板，你需要至少有一个已保存的 visualization。

### 创建一个新的仪表板

你第一次点击 **Dashboard** 标签的时候，Kibana 会显示一个空白的仪表板



通过添加可视化的方式来构建你的仪表板。

### 添加可视化到仪表板上

要添加可视化到仪表板上，点击工具栏面板上的 **Add Visualization** 按钮。从列表中选择一个已保存的可视化。你可以在 **Visualization Filter** 里输入字符串来过滤想要找的可视化。

由你选择的这个可视化会出现在你仪表板上的一个容器(container)里。

如果你觉得容器的高度和宽度不合适，可以[调整容器大小](#)。

## 保存仪表板

要保存仪表板，点击工具栏面板上的 **Save Dashboard** 按钮，在 **Save As** 栏输入仪表板的名字，然后点击 **Save** 按钮。

## 加载已保存仪表板

点击 **Load Saved Dashboard** 按钮显示已存在的仪表板列表。已保存仪表板选择器包括了一个文本栏可以通过仪表板的名字做过滤，还有一个链接到 **Object Editor** 而已管理你的已保存仪表板。你也可以直接点击 **Settings > Edit Saved Objects** 来访问 **Object Editor**。

## 分享仪表板

你可以分享仪表板给其他用户。可以直接分享 Kibana 的仪表板链接，也可以嵌入到你的网页里。

用户必须有 Kibana 的访问权限才能看到嵌入的仪表板。

点击 **Share** 按钮显示 HTML 代码，就可以嵌入仪表板到其他网页里。还带有一个指向仪表板的链接。点击复制按钮  可以复制代码，或者链接到你的黏贴板。

## 嵌入仪表板

要嵌入仪表板，从 Share 页里复制出嵌入代码，然后粘贴进你外部网页应用内即可。

# 定制仪表板元素

---

仪表板里的可视化都存在可以调整大小的容器里。接下来会讨论一下容器。

## 移动容器

点击并按住容器的顶部，就可以拖动容器到仪表板任意位置。其他容器会在必要的时候自动移动，给你在拖动的这个容器空出位置。松开鼠标，容器就会固定在当前停留位置。

## 改变容器大小

移动光标到容器的右下角，等光标变成指向拐角的方向，点击并按住鼠标，拖动改变容器的大小。松开鼠标，容器就会固定成当前大小。

## 删除容器

点击容器右上角的 x 图标删除容器。从仪表板删除容器，并不会同时删除掉容器里用到的已存可视化。

## 查看详细信息

要显示可视化背后的原始数据，点击容器地步的条带。可视化会被有关原始数据详细信息的几个标签替换掉。如下所示：

表格(Table)。底层数据的分页展示。你可以通过点击每列顶部的方式给该列数据排序。

NYCTA: Injury count by type Manhattan

**Table Request Response Statistics**

**filters** **Count**

number_of_cyclist_injured:[1 TO *]	25
number_of_motorist_injured:[1 TO *]	41
number_of_pedestrians_injured:[1 TO *]	58
number_of_persons_injured:[1 TO *]	122

Export: [Raw](#) [Formatted](#)

Page Size

请求(Request)。发送到服务器的原始请求，以 JSON 格式展示。

NYCTA: Injury count by type Manhattan

Table Request Response Statistics

Elasticsearch request body

```
{  
  "size": 0,  
  "aggs": {  
    "2": {  
      "filters": {  
        "filters": {  
          "number_of_cyclist_injured:[1 TO *]": {  
            "query": {  
              "query_string": {  
                "query": "number_of_cyclist_injured:[1 TO *]",  
                "analyze_wildcard": true  
              }  
            }  
          },  
          "number_of_motorist_injured:[1 TO *]": {  
            "query": {  
              "query_string": {  
                "query": "number_of_motorist_injured:[1 TO *]",  
                "analyze_wildcard": true  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
},
```

响应(Response)。从服务器返回的原始响应，以 JSON 格式展示。

NYCTA: Injury count by type Manhattan

Table Request Response Statistics

Elasticsearch response body

```
{  
  "took": 32,  
  "timed_out": false,  
  "_shards": {  
    "total": 5,  
    "successful": 5,  
    "failed": 0  
  },  
  "hits": {  
    "total": 947,  
    "max_score": 0,  
    "hits": []  
  },  
  "aggregations": {  
    "2": {  
      "buckets": {  
        "number_of_cyclist_injured:[1 TO *)": {  
          "doc_count": 25  
        },  
        "number_of_motorist_injured:[1 TO *)": {  
          "doc_count": 41  
        }  
      }  
    }  
  }  
}
```

统计值(Statistics)。和请求响应相关的一些统计值，以数据网格的方式展示。数据报告，请求时间，响应时间，返回的记录条目数，匹配请求的索引模式(index pattern)。

NYCTA: Injury count by type Manhattan

Table Request Response Statistics

Query Duration	32ms
Request Duration	289ms
Hits	947
Index	"nyc_visionzero"

## 修改可视化

点击容器右上角的 *Edit* 按钮  在 [Visualize](#) 页打开可视化编辑。

要使用 Kibana，你就得告诉它你想要探索的 Elasticsearch 索引是那些，这就要配置一个或者更多的索引模式。此外，你还可以：

- 创建脚本化字段，这个字段可以实时从你的数据中计算出来。你可以浏览这种字段，并且在它基础上做可视化，但是不能搜索这种字段。
- 设置高级选项，比如表格里显示多少行，常用字段显示多少个。修改高级选项的时候要千万小心，因为一个设置很可能跟另一个设置是不兼容的。
- 为生产环境配置 Kibana。

## 创建一个连接到 Elasticsearch 的索引模式

一个索引模式定义了一个或者多个你打算探索的 Elasticsearch 索引。Kibana 会查找匹配指定模式的索引名。模式中的通配符()匹配零到多个字符。比如，模式 `myindex-` 匹配所有名字以 myindex- 开头的索引，比如 myindex-1 和 myindex-2`。

如果你用了事件时间来创建索引名(比如说，如果你是用 Logstash 往 Elasticsearch 里写数据)，索引模式里也可以匹配一个日期格式。在这种情况下，模式的静态文本部分必须用中括号包含起来，日期格式能用的字符，参见表 1 "日期格式码"。

比如，`[logstash-]YYYY.MM.DD` 匹配所有名字以 `logstash-` 为前缀，后面跟上 `YYYY.MM.DD` 格式时间戳的索引，比如 `logstash-2015.01.31` 和 `logstash-2015-02-01`。

索引模式也可以简单的设置为一个单独的索引名字。

要创建一个连接到 Elasticsearch 的索引模式：

1. 切换到 `Settings > Indices` 标签页。
2. 指定一个能匹配你的 Elasticsearch 紴引名的索引模式。默认的，Kibana 会假设你是要处理 Logstash 导入的数据。

当你在顶层标签页之间切换的时候，Kibana 会记住你之前停留的位置。比如，如果你在 `Settings` 标签页查看了一个索引模式，然后切换到 `Discover` 标签，再切换回 `Settings` 标签，Kibana 还会显示上次你查看的索引模式。要看到创建模式的表单，需要从索引模式列表里点击 `Add` 按钮。

1. 如果你索引有时间戳字段打算用来做基于事件的对比，勾选 `Index contains time-based events` 然后选择包含了时间戳的索引字段。Kibana 会读取索引映射，列出所有包含了时间戳的字段供选择。
2. 如果新索引是周期性生成，名字里有时间戳的，勾选 `use event times to create index names` 和 `Index pattern interval` 选项。这会让 Kibana 只搜索哪些包含了你指定的时间范围内的数据的索引。当你使用 Logstash 往 Elasticsearch 写数据的时候非常有用。
3. 点击 `Create` 添加索引模式。
4. 要设置新模式作为你查看 `Discover` 页是的默认模式，点击 `favorite` 按钮。

表 1. 日期格式码

格式	描述
M	Month - cardinal: 1 2 3 ... 12
Mo	Month - ordinal: 1st 2nd 3rd ... 12th
MM	Month - two digit: 01 02 03 ... 12
MMM	Month - abbreviation: Jan Feb Mar ... Dec
MMMM	Month - full: January February March ... December
Q	Quarter: 1 2 3 4
D	Day of Month - cardinal: 1 2 3 ... 31
Do	Day of Month - ordinal: 1st 2nd 3rd ... 31st
DD	Day of Month - two digit: 01 02 03 ... 31
DDD	Day of Year - cardinal: 1 2 3 ... 365

DDDo	Day of Year - ordinal: 1st 2nd 3rd ... 365th
DDDD	Day of Year - three digit: 001 002 ... 364 365
d	Day of Week - cardinal: 0 1 3 ... 6
do	Day of Week - ordinal: 0th 1st 2nd ... 6th
dd	Day of Week - 2-letter abbreviation: Su Mo Tu ... Sa
ddd	Day of Week - 3-letter abbreviation: Sun Mon Tue ... Sat
dddd	Day of Week - full: Sunday Monday Tuesday ... Saturday
e	Day of Week (locale): 0 1 2 ... 6
E	Day of Week (ISO): 1 2 3 ... 7
w	Week of Year - cardinal (locale): 1 2 3 ... 53
wo	Week of Year - ordinal (locale): 1st 2nd 3rd ... 53rd
ww	Week of Year - 2-digit (locale): 01 02 03 ... 53
W	Week of Year - cardinal (ISO): 1 2 3 ... 53
Wo	Week of Year - ordinal (ISO): 1st 2nd 3rd ... 53rd
WW	Week of Year - two-digit (ISO): 01 02 03 ... 53
YY	Year - two digit: 70 71 72 ... 30
YYYY	Year - four digit: 1970 1971 1972 ... 2030
gg	Week Year - two digit (locale): 70 71 72 ... 30
gggg	Week Year - four digit (locale): 1970 1971 1972 ... 2030
GG	Week Year - two digit (ISO): 70 71 72 ... 30
GGGG	Week Year - four digit (ISO): 1970 1971 1972 ... 2030
A	AM/PM: AM PM
a	am/pm: am pm
H	Hour: 0 1 2 ... 23
HH	Hour - two digit: 00 01 02 ... 23
h	Hour - 12-hour clock: 1 2 3 ... 12
hh	Hour - 12-hour clock, 2 digit: 01 02 03 ... 12
m	Minute: 0 1 2 ... 59
mm	Minute - two-digit: 00 01 02 ... 59
s	Second: 0 1 2 ... 59
ss	Second - two-digit: 00 01 02 ... 59
S	Fractional Second - 10ths: 0 1 2 ... 9
SS	Fractional Second - 100ths: 0 1 ... 98 99
SSS	Fractional Seconds - 1000ths: 0 1 ... 998 999
Z	Timezone - zero UTC offset (hh:mm format): -07:00 -06:00 -05:00 .. +07:00
ZZ	Timezone - zero UTC offset (hhmm format): -0700 -0600 -0500 ... +0700
X	Unix Timestamp: 1360013296
x	Unix Millisecond Timestamp: 1360013296123

## 设置默认索引模式

---

默认索引模式会在你查看 **Discover** 标签的时候自动加载。Kibana 会在 **Settings > Indices** 标签页的索引模式列表里，给默认模式左边显示一个星号。你创建的第一个模式会自动被设置为默认模式。

要设置一个另外的模式为默认索引模式：

1. 进入 `Settings > Indices` 标签页。
2. 在索引模式列表里选择你打算设置为默认值的模式。
3. 点击模式的 `Favorite` 标签。

你也可以在 `Advanced > Settings` 里设置默认索引模式。

## 重加载索引的字段列表

当你添加了一个索引映射，Kibana 自动扫描匹配模式的索引以显示索引字段。你可以重加载索引字段列表，以显示新添加的字段。

重加载索引字段列表，也会重设 Kibana 的常用字段计数器。这个计数器是跟踪你在 Kibana 里常用字段，然后来排序字段列表的。

要重加载索引的字段列表：

1. 进入 `Settings > Indices` 标签页。
2. 在索引模式列表里选择一个索引模式。
3. 点击模式的 `Reload` 按钮。

## 删除一个索引模式

要删除一个索引模式：

1. 进入 `Settings > Indices` 标签页。
2. 在索引模式列表里选择你打算删除的模式。
3. 点击模式的 `Delete` 按钮。
4. 确认你是想要删除这个索引模式。

## 创建一个脚本化字段

脚本化字段从你的 Elasticsearch 索引数据中即时计算得来。在 **Discover** 标签页，脚本化字段数据会作为文档数据的一部分显示，而且你还可以在可视化里使用脚本化字段。(脚本化字段的值是在请求的时候计算的，所以它们没有被索引，不能搜索到)

即时计算脚本化字段非常消耗资源，会直接影响到 Kibana 的性能。而且记住，Elasticsearch 里没有内置对脚本化字段的验证功能。如果你的脚本有 bug，你会在查看动态生成的数据时看到 exception。

脚本化字段使用 Lucene 表达式语法。更多细节，请阅读 [Lucene Expressions Scripts](#)。

你可以在表达式里引用任意单个数值类型字段，比如：

```
doc['field_name'].value
```

要创建一个脚本化字段：

1. 进入 `Settings > Indices`
2. 选择你打算添加脚本化字段的索引模式。

3. 进入模式的 `Scripted Fields` 标签。
4. 点击 `Add Scripted Field`。
5. 输入脚本化字段的名字。
6. 输入用来即时计算数据的表达式。
7. 点击 `Save Scripted Field`。

有关 Elasticsearch 的脚本化字段的更多细节，阅读 [Scripting](#)。

## 更新一个脚本化字段

---

要更新一个脚本化字段：

1. 进入 `Settings > Indices`。
2. 点击你要更新的脚本化字段的 `Edit` 按钮。
3. 完成变更后点击 `Save Scripted Field` 升级。

注意 Elasticsearch 里没有内置对脚本化字段的验证功能。如果你的脚本有 bug，你会在查看动态生成的数据时看到 exception。

## 删除一个脚本化字段

---

要删除一个脚本化字段：

1. 进入 `Settings > Indices`。
2. 点击你要删除的脚本化字段的 `Delete` 按钮。
3. 确认你确实想删除它。

## 设置高级参数

---

高级参数页允许你直接编辑那些控制着 Kibana 应用行为的设置。比如，你可以修改显示日期的格式，修改默认的索引模式，设置十进制数值的显示精度。

修改高级参数可能带来意想不到的后果。如果你不确定自己在做什么，最好离开这个设置页面。

要设置高级参数：

1. 进入 `Settings > Advanced`。
2. 点击你要修改的选项的 `Edit` 按钮。
3. 给这个选项输入一个新的值。
4. 点击 `Save` 按钮。

## 管理已保存的搜索，可视化和仪表板

---

你可以从 **Settings > Objects** 查看，编辑，和删除已保存的搜索，可视化和仪表板。

查看一个已保存的对象会显示在 **Discover**, **Visualize** 或 **Dashboard** 页里已选择的项目。要查看一个已保存对象：

1. 进入 `Settings > Objects`。
2. 选择你想查看的对象。
3. 点击 `View` 按钮。

编辑一个已保存对象让你可以直接修改对象定义。你可以修改对象的名字，添加一段说明，以及修改定义这个对象的属性的 JSON。

如果你尝试访问一个对象，而它关联的索引已经被删除了，Kibana 会显示这个对象的编辑(Edit Object)页。你可以：

- 重建索引这样就可以继续用这个对象。
- 删除对象，然后用另一个索引重建对象。
- 在对象的 `kibanaSavedObjectMeta.searchSourceJSON` 里修改引用的索引名，指向一个还存在的索引模式。这个在你的索引被重命名了的情况下非常有用。

对象属性没有验证机制。提交一个无效的变更会导致对象不可用。通常来说，你还是应该用 Discover, Visualize 或 Dashboard 页面来创建新对象而不是直接编辑已存在的对象。

要编辑一个已保存的对象：

1. 进入 `Settings > Objects`。
2. 选择你想编辑的对象。
3. 点击 `Edit` 按钮。
4. 修改对象定义。
5. 点击 `Save object` 按钮。

要删除一个已保存的对象：

1. 进入 `Settings > Objects`。
2. 选择你想删除的对象。
3. 点击 `Delete` 按钮。
4. 确认你确实想删除这个对象。

## 设置 kibana 服务器属性

Kibana 服务器在启动的时候会从 `kibana.yml` 文件读取属性设置。默认设置是运行在 `localhost:5601`。要变更主机或端口，或者连接远端主机上的 Elasticsearch，你都需要更新你的 `kibana.yml` 文件。你还可以开启 SSL 或者设置其他一系列选项：

表 2. Kibana 服务器属性

属性	描述
<code>port</code>	Kibana 服务器运行的端口。默认： <code>port: 5601</code> 。
<code>host</code>	Kibana 服务器监听的地址。默认： <code>host: "0.0.0.0"</code> 。
<code>elasticsearch_url</code>	你想请求的索引存在哪个 Elasticsearch 实例上。默认： <code>elasticsearch_url: "http://localhost:9200"</code> 。
<code>elasticsearch_preserve_host</code>	默认的，浏览器请求中的主机名即作为 Kibana 发送给 Elasticsearch 时请求的主机名。如果你设置这个参数为 <code>false</code> ，Kibana 会改用 <code>elasticsearch_url</code> 里的主机名。你应该不用担心这个设置——直接用默认即可。默认： <code>elasticsearch_preserve_host: true</code> 。
<code>kibana_index</code>	保存搜索，可视化，仪表板信息的索引的名字。默认： <code>kibana_index: .kibana</code> 。
<code>default_app_id</code>	进入 Kibana 是默认显示的页面。可以为 <code>discover</code> , <code>visualize</code> , <code>dashboard</code> 或 <code>settings</code> 。默认： <code>default_app_id: "discover"</code> 。
<code>request_timeout</code>	等待 Kibana 后端或 Elasticsearch 的响应的超时时间，单位毫秒。默认： <code>request_timeout: 500000</code> 。
<code>shard_timeout</code>	Elasticsearch 等待分片响应的超时时间。设置为 0 表示关闭超时控制。默认： <code>shard_timeout: 0</code> 。
<code>verify_ssl</code>	定义是否验证 Elasticsearch SSL 证书。设置为 <code>false</code> 关闭 SSL 认证。默认： <code>verify_ssl: true</code> 。
<code>ca</code>	你的 Elasticsearch 实例的 CA 证书的路径。如果你是自己签的证书，必须指定这个参数，证书才能被认证。否则，你需要关闭 <code>verify_ssl</code> 。默认： <code>none</code> 。
<code>ssl_key_file</code>	Kibana 服务器的密钥文件路径。设置用来加密浏览器和 Kibana 之间的通信。默认： <code>none</code> 。

ssl_cert_file	Kibana 服务器的证书文件路径。设置用来加密浏览器和 Kibana 之间的通信。默认 : none。
pid_file	你想用来存进程 ID 文件的位置。如果没有指定, PID 文件存在 /var/run/kibana.pid 。默认 : none。

当你准备在生产环境使用 Kibana 的时候，比起在本机运行，你需要多考虑一些：

- 在哪运行 kibana
- 是否需要加密 Kibana 出入的流量
- 是否需要控制访问数据的权限

## 部署的考虑

你怎么部署 Kibana 取决于你的运用场景。如果就是自己用，在本机运行 Kibana 然后配置一下指向到任意你想交互的 Elasticsearch 实例即可。如果你有一大批 Kibana 重度用户，可能你需要部署多个 Kibana 实例，指向同一个 Elasticsearch，然后前面加一个负载均衡。

虽然 Kibana 不是资源密集型的应用，我们依然建议你单独用一个节点来运行 Kibana，而不是泡在 Elasticsearch 节点上。

## 配置 Kibana 和 shield 一起工作

如果你在用 Shield 做 Elasticsearch 用户认证，你需要给 Kibana 提供用户凭证，这样它才能访问 `.kibana` 索引。Kibana 用户需要由权限访问 `.kibana` 索引里以下操作：

```
'.kibana':  
  - indices:admin/create  
  - indices:admin/exists  
  - indices:admin/mapping/put  
  - indices:admin/mappings/fields/get  
  - indices:admin/refresh  
  - indices:admin/validate/query  
  - indices:data/read/get  
  - indices:data/read/mget  
  - indices:data/read/search  
  - indices:data/write/delete  
  - indices:data/write/index  
  - indices:data/write/update  
  - indices:admin/create
```

更多配置 Shield 的内容，请阅读 [Shield with Kibana 4](#)。

要配置 Kibana 的凭证，设置 `kibana.yml` 里的 `kibana_elasticsearch_username` 和 `kibana_elasticsearch_password` 选项即可：

```
# If your Elasticsearch is protected with basic auth:  
kibana_elasticsearch_username: kibana4  
kibana_elasticsearch_password: kibana4
```

## 开启 ssl

Kibana 同时支持对客户端请求以及 Kibana 服务器发往 Elasticsearch 的请求做 SSL 加密。

要加密浏览器到 Kibana 服务器之间的通信，配置 `kibana.yml` 里的 `ssl_key_file` 和 `ssl_cert_file` 参数：

```
# SSL for outgoing requests from the Kibana Server (PEM formatted)  
ssl_key_file: /path/to/your/server.key  
ssl_cert_file: /path/to/your/server.crt
```

如果你在用 Shield 或者其他提供 HTTPS 的代理服务器保护 Elasticsearch，你可以配置 Kibana 通过 HTTPS 方式访问 Elasticsearch，这样 Kibana 服务器和 Elasticsearch 之间的通信也是加密的。

要做到这点，你需要在 `kibana.yml` 里配置 Elasticsearch 的 URL 时指明是 HTTPS 协议：

```
elasticsearch: "https://<your_elasticsearch_host>.com:9200"
```

如果你给 Elasticsearch 用的是自己签名的证书，请在 `kibana.yml` 里设定 `ca` 参数指明 PEM 文件位置，这也意味着开启了 `verify_ssl` 参数：

```
# If you need to provide a CA certificate for your Elasticsearch instance, put  
# the path of the pem file here.  
ca: /path/to/your/ca/cacert.pem
```

## 控制访问权限

---

你可以用 [Elasticsearch Shield](#) 来控制用户通过 Kibana 可以访问到的 Elasticsearch 数据。Shield 提供了索引级别的访问控制。如果一个用户没被许可运行这个请求，那么它在 Kibana 可视化界面上只能看到一个空白。

要配置 Kibana 使用 Shield，你要为 Kibana 创建一个或者多个 Shield 角色(role)，以 `kibana4` 作为开头的默认角色。更详细的做法，请阅读 [Using Shield with Kibana 4](#)。

Kibana 4 提供了一系列新特性，让你在找问题，解决问题的时候前所未有的简单。它有一个全新的界面，而且优化了搜索和展示数据，构建和分析仪表板的工作流程。

## 关键特性

---

- 全新的数据搜索和发现界面
- 统一的可视化构建器，用以构建你喜欢和新加入的那些图表：
  - 区块图
  - 数据表格
  - 折线图
  - Markdown 文本挂件
  - 饼图(包括甜圈图)
  - 原始文档挂件
  - 单数值挂件
  - 贴片地图
  - 垂直柱状图
- 可拖拽的仪表板构建方式，让你可以快速添加，删除可视化，已经修改其大小和长宽比
- 基于聚合接口的高级分析能力，现在支持：
  - 去重统计(cardinality)
  - 非时间的直方图
  - 范围统计(Ranges)
  - 关键词(Significant terms)
  - 百分比(Percentiles)
- 基于 Lucene Expressions 的脚本化字段让你可以完成临时计算任务

## 优化

---

- 可以保存搜索和可视化，这样你可以再多个仪表板上链接和使用同一个搜索构建的可视化
- 可视化支持不限次数的层叠聚合，这样你可以生成新的可视化样式，比如双层甜圈图
- 替换模板化和脚本化仪表板需求的新 URL 格式
- 更好的移动端体验
- 更快的仪表板加载速度，因为我们消减了发出的 HTTP 请求数量
- 客户端请求和发往 Elasticsearch 的请求都可以有 SSL 加密
- 搜索结果高亮
- 可以很容易访问和导出可视化背后的数据：
  - 可以在表格中查看，也可以用 JSON 格式
  - 导出成 CSV 格式
  - 查看 Elasticsearch 请求和响应
- 跟仪表板一样，可以分享和嵌入独立的可视化部分

## 其他细节

---

- 自带网页服务器，用 Node.js 作为后端——发布有 Linux, Windows 和 Mac OS 的二进制文件分发
- 用 D3 框架做可视化效果

# 源码剖析

---

Kibana 4 采用 angular.js + node.js 框架编写。其中 node.js 主要提供两部分功能，给 Elasticsearch 做搜索请求转发代理，以及 auth、ssl、setting 等操作的服务器后端。

本章节假设你已经对 angular 有一定程度了解——至少是阅读并理解了 kibana 3 源码剖析章节内容的程度。所以不会再解释其中 angular 的 route, controller, directive, service, factory 等概念。

如果打算迁移 kibana 3 的 CAS 验证功能到 kibana 4，那么可以稍微了解一下 `index.js`, `app.js`, `lib/auth.js` 里的 `htpasswd` 简单实现，相信可以很快修改成功。本章主要还是集中在前端 kibana 页面功能的实现上。

在 Elastic{ON} 大会上，也有专门针对 Kibana 4 源码和二次开发入门的演讲。请参阅：<https://speakerdeck.com/elastic/the-contributors-guide-to-the-kibana-galaxy>

# 主页入口

---

kibana 4 主页入口，分析方法跟 kibana 3 一样，看 index.html 和 require.config.js 即可。由此可以看到，首先进入的，应该是 index.js。index.js 根据 configFile 设置默认 routes，然后执行 kibana.load() 函数，首先加载 plugins/kibana/index.js，然后加载其他 plugins。

设置 routes 的具体操作在加载的 utils/routes/index.js 文件里，其中调用 utils/routes/\_setup.js，在未设置 default index pattern 的时候跳转 URL 到 "/settings/indices" 页面。

plugins/kibana/index.js 里又有一些列操作：首先加载 components/setup/setup.js 和 components/config/config.js 两个 angular.service，然后加载 plugins/kibana/\_init，plugins/kibana/\_apps，plugins/kibana/\_timepicker。

## setup 过程

---

components/setup/setup.js 依次调用 components/setup/steps/ 下的 check\_for\_es，check\_es\_version，check\_for\_kibana\_index，如果没有 kibana index，再调用一个 create\_kibana\_index。完成。

components/config/config.js 主要是从 kibana index 里的 "config" type 中读取 "kbnVersion" id 的数据。这个 "kbnVersion" 是源代码(index.js)里的一个常量，在 grunt 编译时会生成的。

plugins/kibana/\_init 里监听 application.load 事件，触发 courses.start() 函数。

plugins/kibana/\_apps 提供路径记忆(lastPath)功能，这点在 kibana4 user guide 上被专门提到过；然后初始化 registry/apps，并循环调用 assignPaths 和 getShow 方法。

plugins/kibana/\_timepicker 提供时间选择器页面。

## courses 概述

---

components/courses/courses.js 中，加载 index\_pattern 和 saved\_objects，启动 searchLooper 和 docLooper；设置整个页面的定期刷新。

courses 是一个非常重要的东西，除了上行提到的这几个以外，目录下还有 docSource 和 searchSource，可以简单理解为 kibana 跟 ES 之间的一个 object mapper。

searchLooper, docLooper 则是限制在 \_request\_queue 里的 Looper 对象，分别给 Looper.start 方法传递 FetchStrategyForSearch, FetchStrategyForDoc，对应 ES 的 /\_msearch 和 /\_mget 请求。这两个在 components/courier/fetch/strategy/search.js 和 components/courier/fetch/strategy/doc.js 里定义。

## registry 概述

---

registry/apps.js 主要是加载 registry/\_registry.js，把注册的 app 存入 utils/indexed\_array/index 的 IndexedArray 对象。对象主要有几个值：id, name, order。前面说到的两个方法，assignPaths 里就是用 app.id 设置 lastPath，而 getShow 里就是用 order 来判断是否展示在页面上。

所以这里就体现出 kibana 4 的可扩展性了。事实上，在服务器端的 index.js 上，就有下面配置：

```
external_plugins_folder : process.env.PLUGINS_FOLDER || null,  
bundled_plugins_folder : path.resolve(public_folder, 'plugins'),
```

可以看到，官方的 apps，都是在 plugins 目录下的，而自己开发的 apps，可以通过环境变量 \$PLUGINS\_FOLDER 设置加载进来。

下一章，我们开始介绍官方提供的几个 apps。

# 搜索页

---

`plugins/discover/index.js` 中主要就是注册自己的 id, name, order 到上节最后说的 `registry.apps` 里。此外就是加载本 app 目录内的其他文件。依次说明如下：

## plugins/discover/saved\_searches/saved\_searches.js

---

- 定义 `savedSearches` 这个 angular service, 用来操作 `kibana_index` 索引里 `search` 这个类型下的数据；
- 加载了 `saved_searches/_saved_searches.js` 提供的 `savedSearch` 这个 angular factory, 这里定义了一个搜索 (search) 在 `kibana_index` 里的数据结构, 包括 title, description, hits, column, sort, version 等字段(这部分内容, 可以直接通过读取 Elasticsearch 中的索引内容看到, 比阅读代码更直接, 本章最后即专门介绍 `kibana_index` 中的数据结构), 然后用前面提到的 `components/couries/saved_object/saved_object.js` 跟索引交互；
- 还加载并注册了 `plugins/settings/saved_object_registry.js`, 表示可以在 settings 里修改这里的 `savedSearches` 对象。

## plugins/discover/directives/timechart.js

---

- 加载 `components/vislib/index.js`。
- 提供 `discoverTimechart` 这个 angular directive, 监听 "data" 并调用 `vislib.Chart` 对象绘图。

## plugins/discover/components/field\_chooser/field\_chooser.js

---

- 提供 `discFieldChooser` 这个 angular directive, 其中监听 "fields" 并调用 `calculateFields` 计算常用字段排行, 监听 "data" 并调用 `$scope.details()` 方法, 提供 `$scope.runAgg()` 方法。方法中, 根据字段的类型不同, 分别可能使用 `date_histogram` / `geohash_grid` / `terms` 聚合函数, 创建可视化模型, 然后带着当前页这些设定(前面说过, 各 app 之间通过 `sessionStorage` 共享了状态的)跳转到 "/visualize/create" 页面, 相当于是这三个常用聚合的快速可视化操作。
- 加载 `plugins/discover/components/field_chooser/lib/field_calculator.js`, 提供 `fieldCalculator.getFieldValueCounts()` 方法, 在 `$scope.details()` 中读取被点击的字段值的情况。
- 加载 `plugins/discover/components/field_chooser/discover_field.js`, 提供 `discoverField` 这个 angular directive, 用于弹出浮层展示零时的 `visualize`(调用上一条提供的 `$scope.details()` 方法), 同时给被点击的字段加常用度；加载 `plugins/discover/components/field_chooser/lib/detail_views/string.html` 网页, 用于浮层效果。网页中对 `indexed` 或 `scripted` 类型的字段, 可以调用前面提到的 `runAgg()` 方法。
- 加载并渲染 `plugins/discover/components/field_chooser/field_chooser.html` 网页。网页中使用了上一条提供的 `discoverField` 标签。

## plugins/discover/controllers/discover.js

---

加载了诸多 js, 主要做了：

- 为 "/discover/:id" 提供 route 并加载 `plugins/discover/index.html` 网页。
- 提供 `discover` 这个 angular controller。
- 加载 `components/vis/vis.js` 并在 `setupVisualization` 函数中绘制 histogram 图。
- 加载 `components/filter_manager/filter_manager.js`, 根据字段类型生成不同的 filter 语句, 存在全局 state 里。
- 加载 `components/doc_table/lib/get_sort.js` (存疑：`docTable` 这个 directive 是在哪里加载的？)

# kibana\_index 结构

---

包括有以下 type :

## config

\_id 为 kibana4 的 version。内容主要是 defaultIndex，设置默认的 index\_pattern.

## search

\_id 为 discover 上保存的搜索名称。内容主要是 title, column, sort 和 kibanaSavedObjectMeta。kibanaSavedObjectMeta 内是一个 searchSourceJSON，保存搜索 json 的字符串。

## visualization

\_id 为 visualize 上保存的可视化名称。内容包括 title, savedSearchId, kibanaSavedObjectMeta 和 visState。其中 visState 里保存了聚合 json 的字符串。如果绑定了已保存的搜索，那么把其在 search 类型里的 \_id 存在 savedSearchId 字段里，如果是从新搜索开始的，那么把搜索 json 的字符串直接存在自己的 kibanaSavedObjectMeta 的 searchSourceJSON 里。

## index-pattern

\_id 为 setting 中设置的 index pattern。内容主要是匹配该模式的所有索引的全部字段与字段映射。如果是基于时间的索引模式，还会有主时间字段 timeFieldName 和时间间隔 intervalName 两个字段。

field 数组中，每个元素是一个字段的情况，包括字段的 type, name, indexed, analyzed, doc\_values, count, scripted 这些状态。

如果 scripted 为 true，那么这个元素就是通过 kibana4 页面添加的脚本化字段，那么这条字段记录还会额外多几个内容：

- script: 记录实际 script 语句。
- lang: 在 Elasticsearch 的 datanode 上采用什么 lang-plugin 运行。默认是 expression。即 ES 1.4.4 开始默认启用的 Lucene expression。在目前的 kibana4 页面上，不提供对这个的修改，所以统一都是这个值。
- type: 因为 Lucene expression 目前只支持对数值型字段做操作，所以目前 kibana4 页面上也不提供对这个的修改，直接默认为 "number"。

对确认要使用其他 lang-plugin 的，目前来说，可以自行修改 kibana\_index 里的 index-pattern 类型中的数据，修改成 "lang": "groovy", "type": "string" 即可。页面上是可以通用的。

# 捐赠者名单

感谢以下童鞋的捐助，让我有动力持续下来并且注册了 <http://kibana.logstash.es> 这么个有意思的域名用来发布本书：

donor	value
颜*	50.00
赵*远	10.00
韩*光	22.00
刘*卫	10.00
185****6322	5.00
彭*源	6.11
余*	100.00
李*灿	20.00
孙*	101.00
彭*根	50.00
史*春	10.00
张*	18.76
陈*林	66.00
吴*维	10.00
周*	10.00
叶*	50.00
张*贝	51.00
刘*	10.00
肖*宇	25.60
周*	100.00
高*达	10.00
庞*	30.00
林*光	10.00
张*	20.00
刘*	10.00
周*文	22.21
俞*	24.80

共计：852.48 元，域名注册已用 275.50 元(3 年)。