

大型架构

NSD ELK

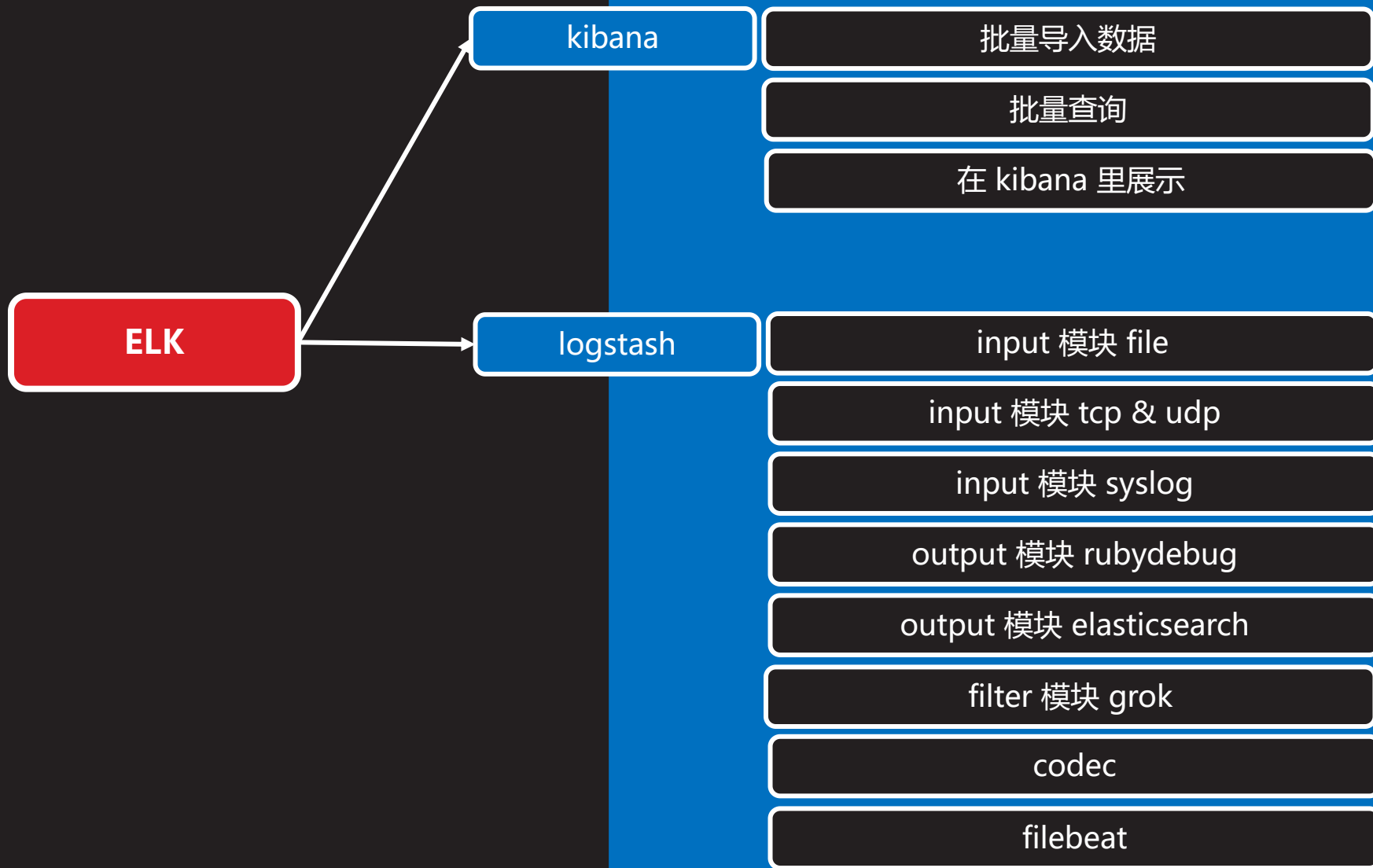
DAY02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	数据批量导入
	10:30 ~ 11:20	kibana使用
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	Logstash安装配置 扩展插件应用
	15:00 ~ 15:50	
	16:10 ~ 17:00	
	17:10 ~ 18:00	总结和答疑



ELK



数据批量导入



数据批量导入

- 使用 `_bulk` 批量导入数据
 - 批量导入数据使用 POST 方式，数据格式为 json，url 编码使用 data-binary
 - 导入含有 index 配置的 json 文件


```
gzip -d logs.jsonl.gz
curl -XPOST 'http://192.168.4.14:9200/_bulk' --data-binary @logs.jsonl

gzip -d shakespeare.json.gz
curl -XPOST 'http://192.168.4.14:9200/_bulk' --data-binary @shakespeare.json
```



数据批量导入

- 使用 `_bulk` 批量导入数据
 - 导入没有有 index 配置的 json 文件
 - 我们需要在 uri 里面制定 index 和 type

```
gzip -d accounts.json.gz
curl -XPOST
'http://192.168.4.14:9200/accounts/act/_bulk?pretty' --
data-binary @accounts.json
```



数据批量查询

- 数据批量查询使用 GET

```
curl -XGET 'http://192.168.4.11:9200/_mget?pretty' -d '{
  "docs":[
    {
      "_index": "accounts",
      "_type": "act",
      "_id": 1
    },
    {
      "_index": "accounts",
      "_type": "act",
      "_id": 2
    },
  ],
}
```



数据批量查询

- 数据批量查询使用 GET
 - 续上一页

```
{  
  "_index": "shakespeare",  
  "_type": "scene",  
  "_id": 1  
}  
]  
'
```



map 映射

- mapping :
 - 映射：创建索引的时候，可以预先定义字段的类型及相关属性。
 - 作用：这样会让索引建立得更加的细致和完善。
 - 分类：静态映射和动态映射。
 - 动态映射：自动根据数据进行相应的映射。
 - 静态映射：自定义字段映射数据类型。



kibana 部分

kibana 部分

- 数据导入以后查看 logs 是否导入成功

logstash-2015.05.20	logstash-2015.05.19	logstash-2015.05.18
size: 27.3Mi (57.7Mi) docs: 4,750 (9,500)	size: 25.4Mi (58.6Mi) docs: 4,624 (9,248)	size: 26.8Mi (58.5Mi) docs: 4,631 (9,262)
信息 动作	信息 动作	信息 动作
0 1 2 3	0 1 2 3	0 1 2 3
4	4	4
0 1 2 3	0 1 2 3	0 1 2 3
4	4	4



kibana 部分

- 修改 kibana 的配置文件后启动 kibana , 然后查看 ES 集群 , 如果出现 .kibana Index 表示 kibana 与 ES 集群连接成功



kibana 部分

- kibana 里选择日志
 - 支持通配符 *
 - 我们这里选择 logstash-*
 - 在下面的 Time-field 选择 @timestamp 作为索引
 - 然后点 create 按钮



kibana 部分

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. They are also used to configure fields.

☒ Index contains time-based events

☐ Use event times to create index names [DEPRECATED]

Index name or pattern

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

logstash-*

☐ Do not expand index pattern when searching (Not recommended)

By default, searches against any time-based index pattern that contains a wildcard will automatically be expanded to query only the indices that contain data within the currently selected time range.

Searching against the index pattern *logstash-** will actually query elasticsearch for the specific matching indices (e.g. *logstash-2015.12.21*) that fall within the current time range.

Time-field name ⓘ refresh fields

Create



kibana 部分

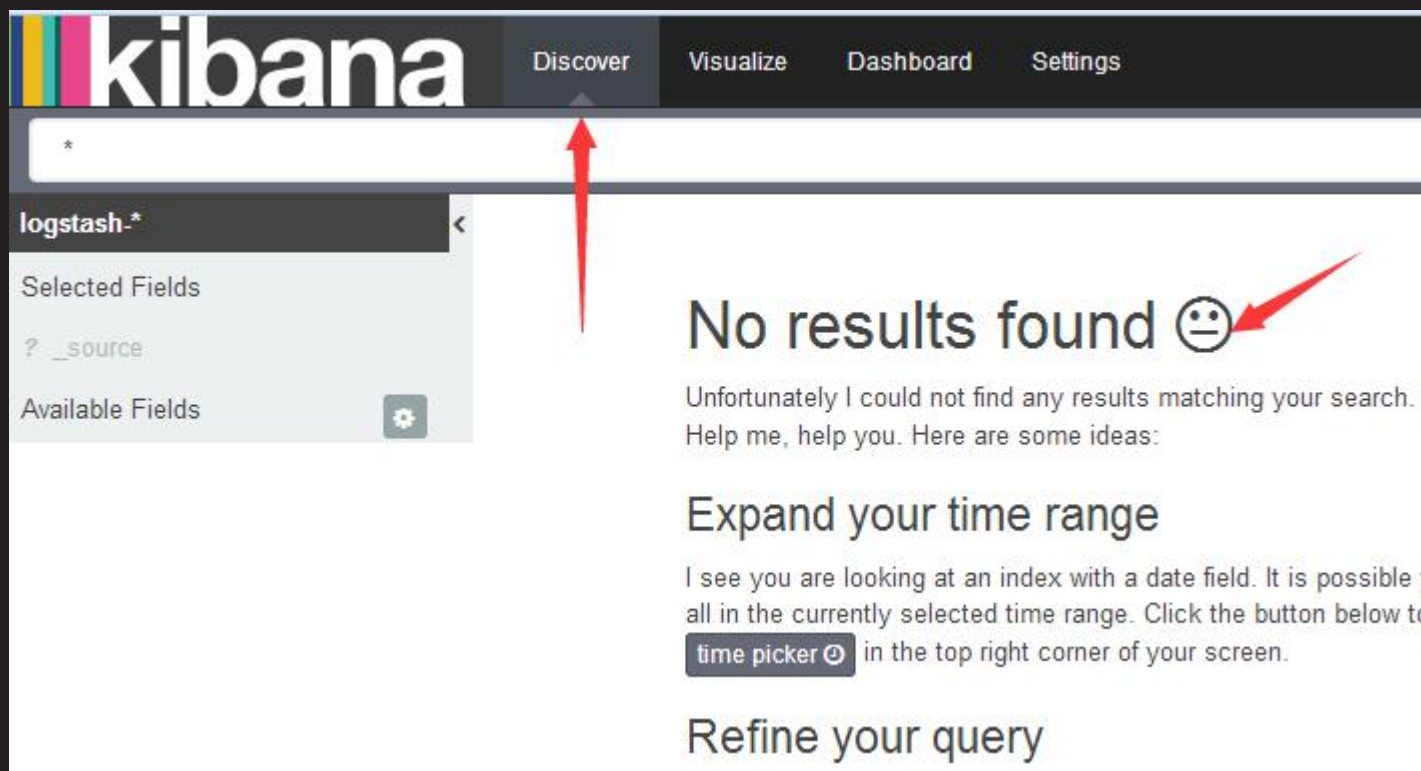
The screenshot shows the Kibana web interface. The top navigation bar includes 'Discover', 'Visualize', 'Dashboard', and 'Settings'. The left sidebar has 'Indices', 'Advanced', 'Objects', 'Status', and 'About'. Under 'Indices', there's a section for 'Index Patterns' with a '+ Add New' button and a list containing '★ logstash-*'. The main content area is titled '★ logstash-*' and includes a description: 'This page lists every field in the logstash-* index and the field's associated core type. Field types must be done using Elasticsearch's Mapping API'. Below this is a 'Filter' input field and two tabs: 'fields (90)' and 'scripted fields (0)'. A table lists fields and their types:

name	type
relatedContent.og:type	string
headings.raw	string
relatedContent.twitter:image	string
utc_time	date
geo.coordinates.lon	number
clientip	string



kibana 部分

- 导入成功以后选择 discover



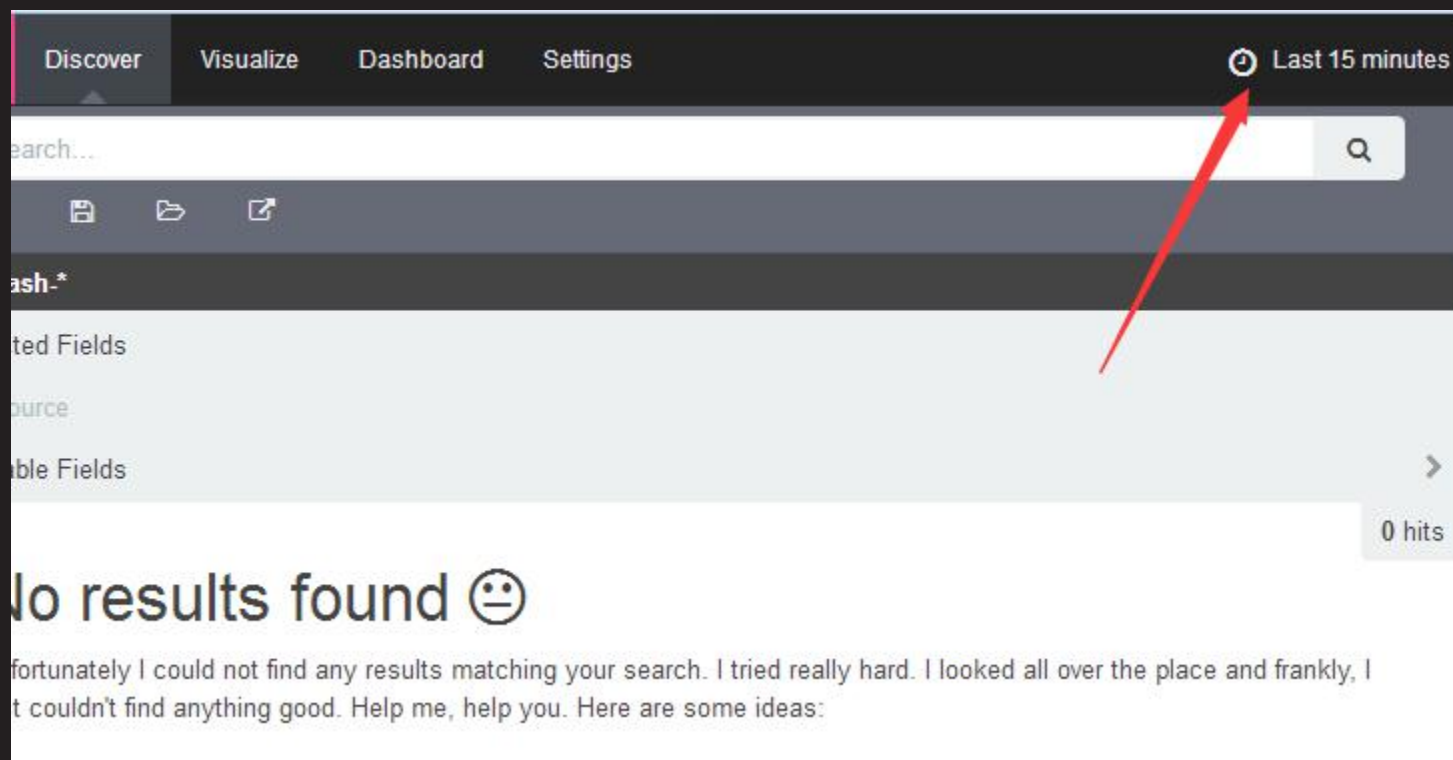
kibana 部分

- 这里没有数据的原因是我们导入的日志是 2015-05-10 至 2015-05-20 时间段的，默认配置是最近 15 分钟，这里我们修改一下时间来显示



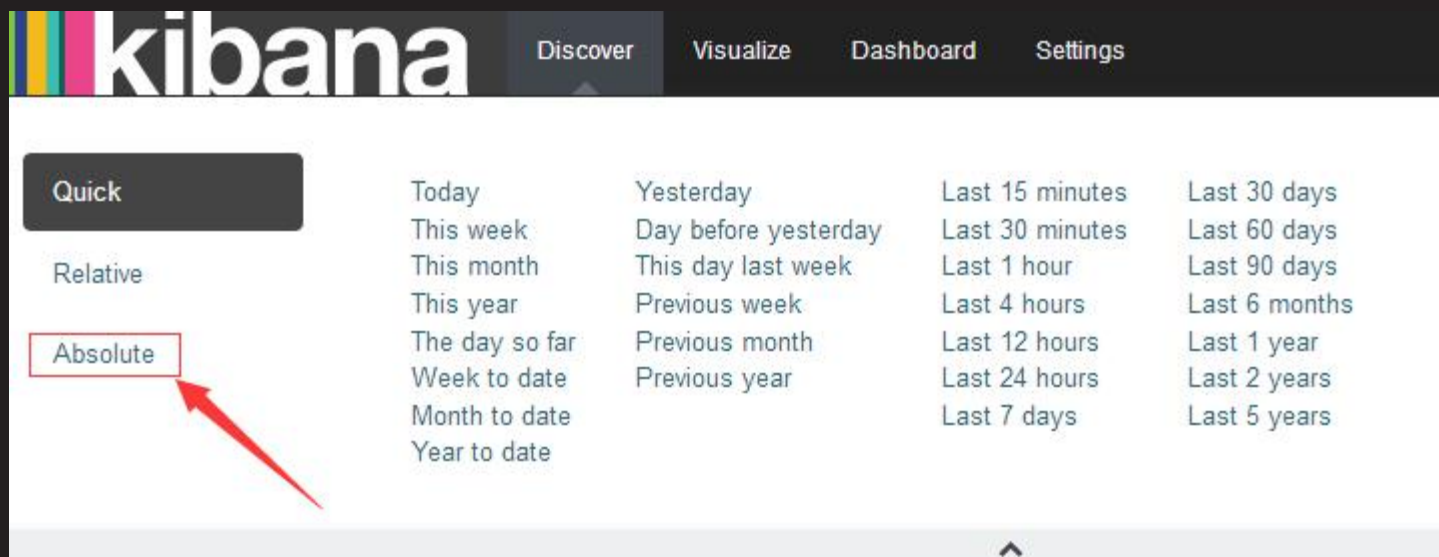
kibana 部分

- 修改时间



kibana 部分

- 修改时间



kibana 部分

- 修改时间

From:

YYYY-MM-DD HH:mm:ss.SSS

<

May 2015

>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	01	02
03	04	05	06	07	08	09
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	01	02	03	04	05	06

To: Set To Now

YYYY-MM-DD HH:mm:ss.SSS

<

May 2015

>

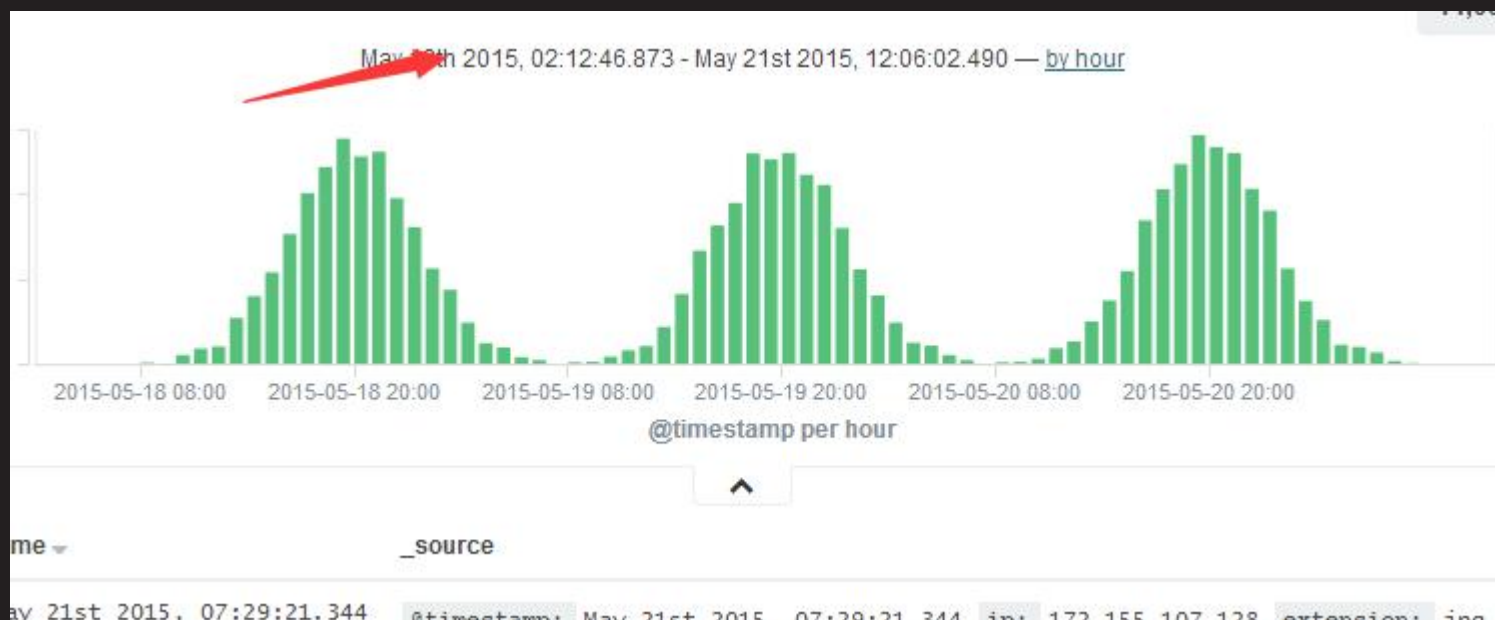
Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28	29	30	01	02
03	04	05	06	07	08	09
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	01	02	03	04	05	06

Go











kibana 部分

- 数据展示



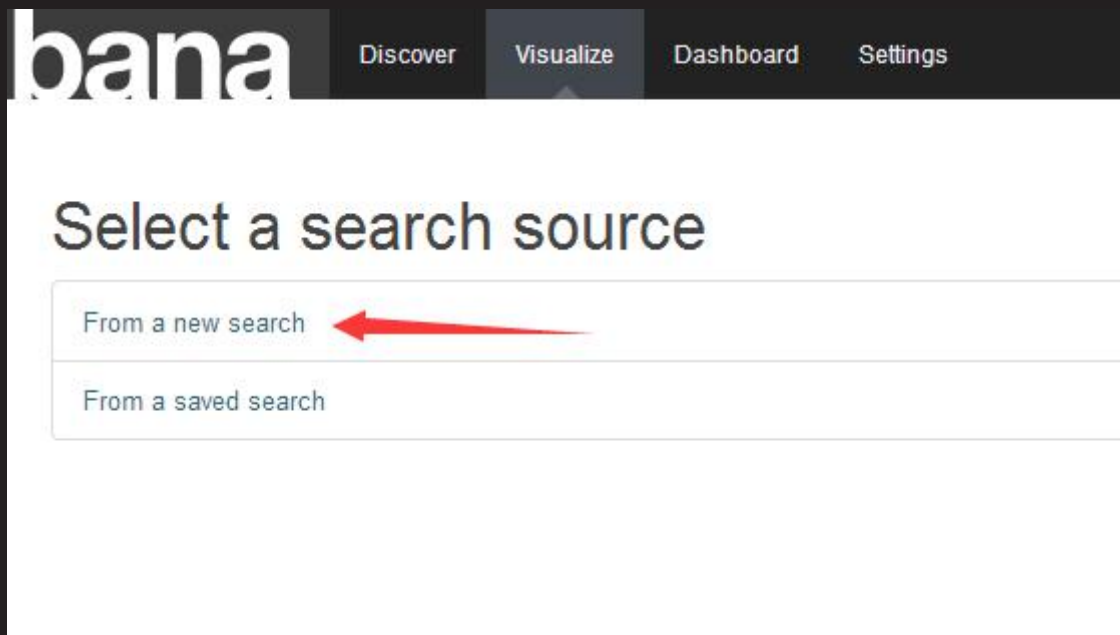
kibana 部分

- 除了柱状图，kibana 还支持很多种展示方式

 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 showing 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 you need, you could do worse than to start here.

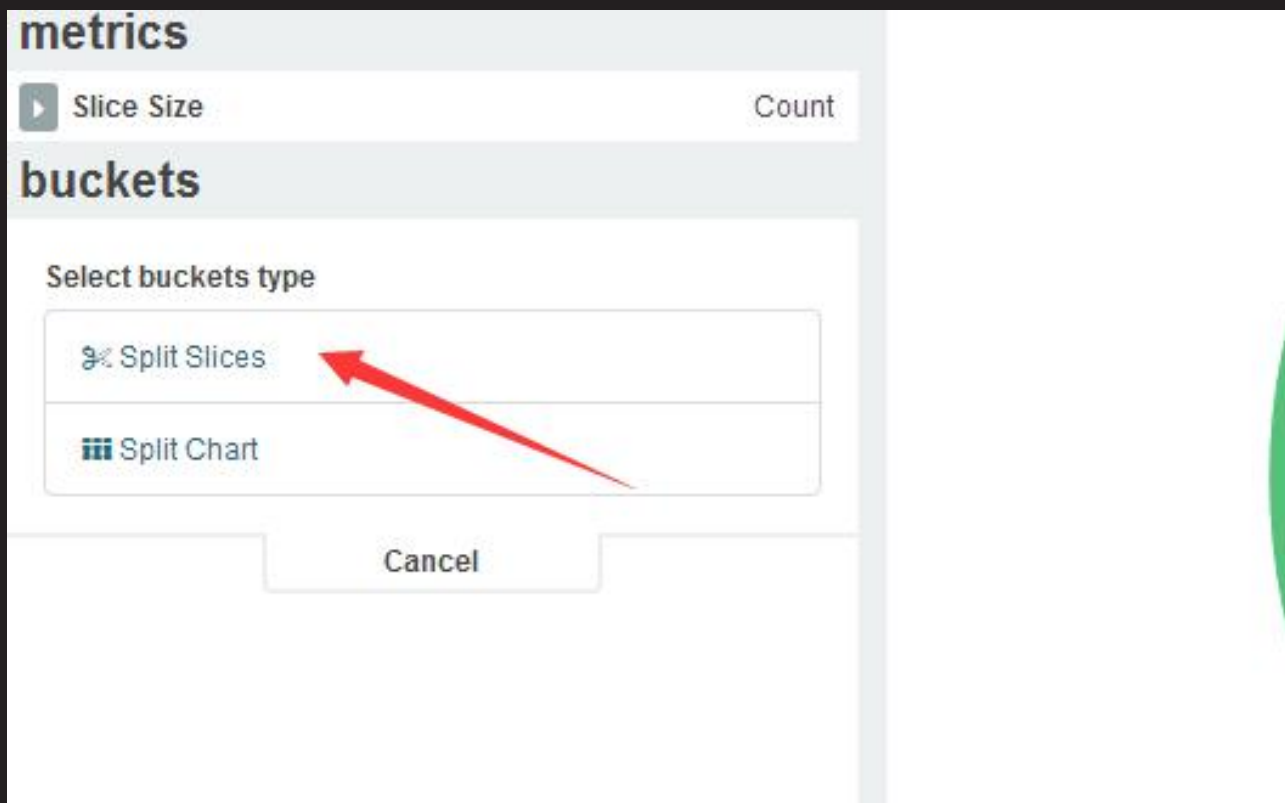
kibana 部分

- 饼图与列表，多种维度自定义统计分析



kibana 部分

- 饼图与列表，多种维度自定义统计分析



kibana 部分

- 饼图与列表，多种维度自定义统计分析

Slice Size Count

buckets

Split Slices

Aggregation

Terms

Field

Order By

metric: Count

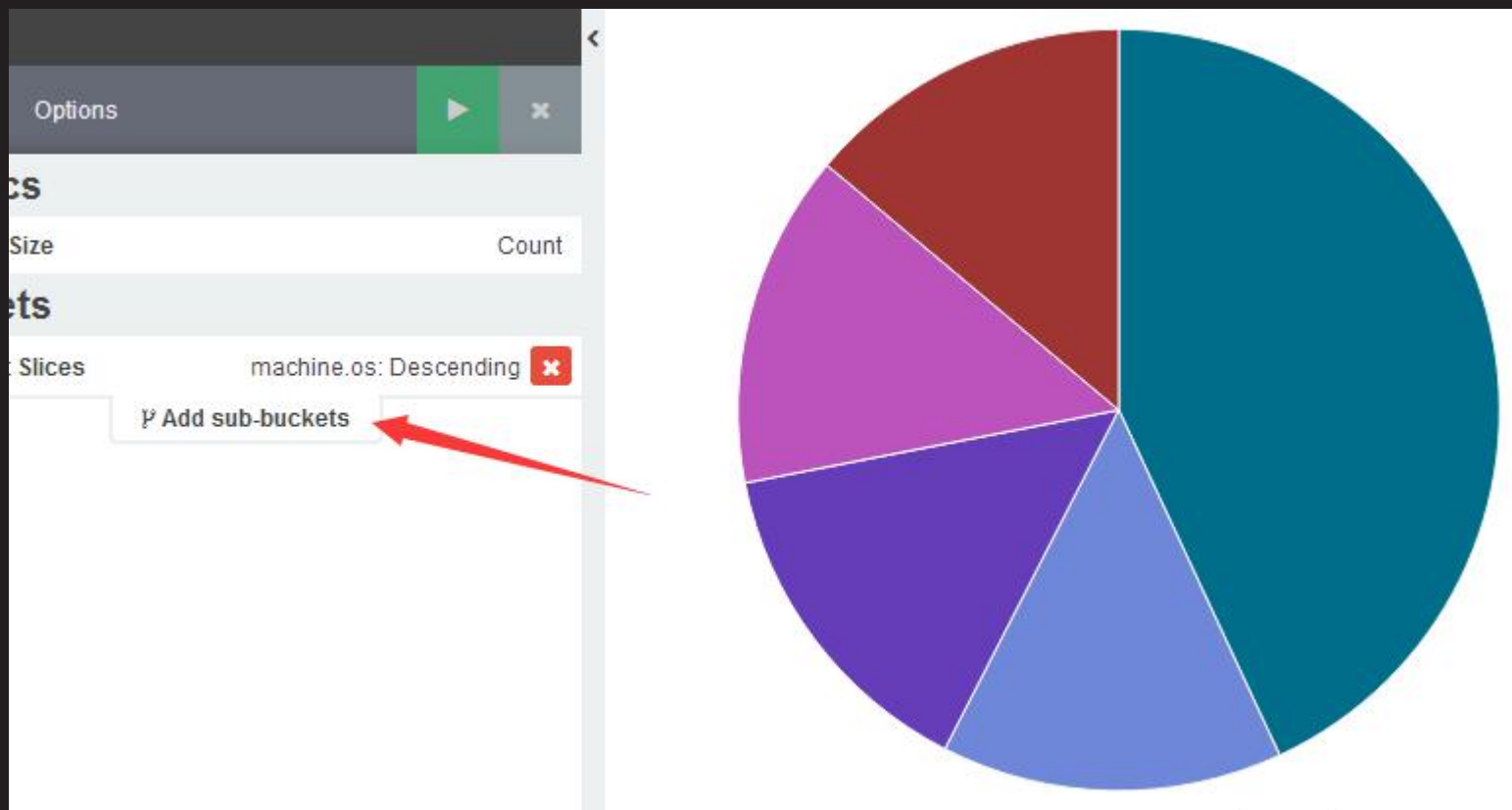
Order Size

Descending 5

CustomLabel

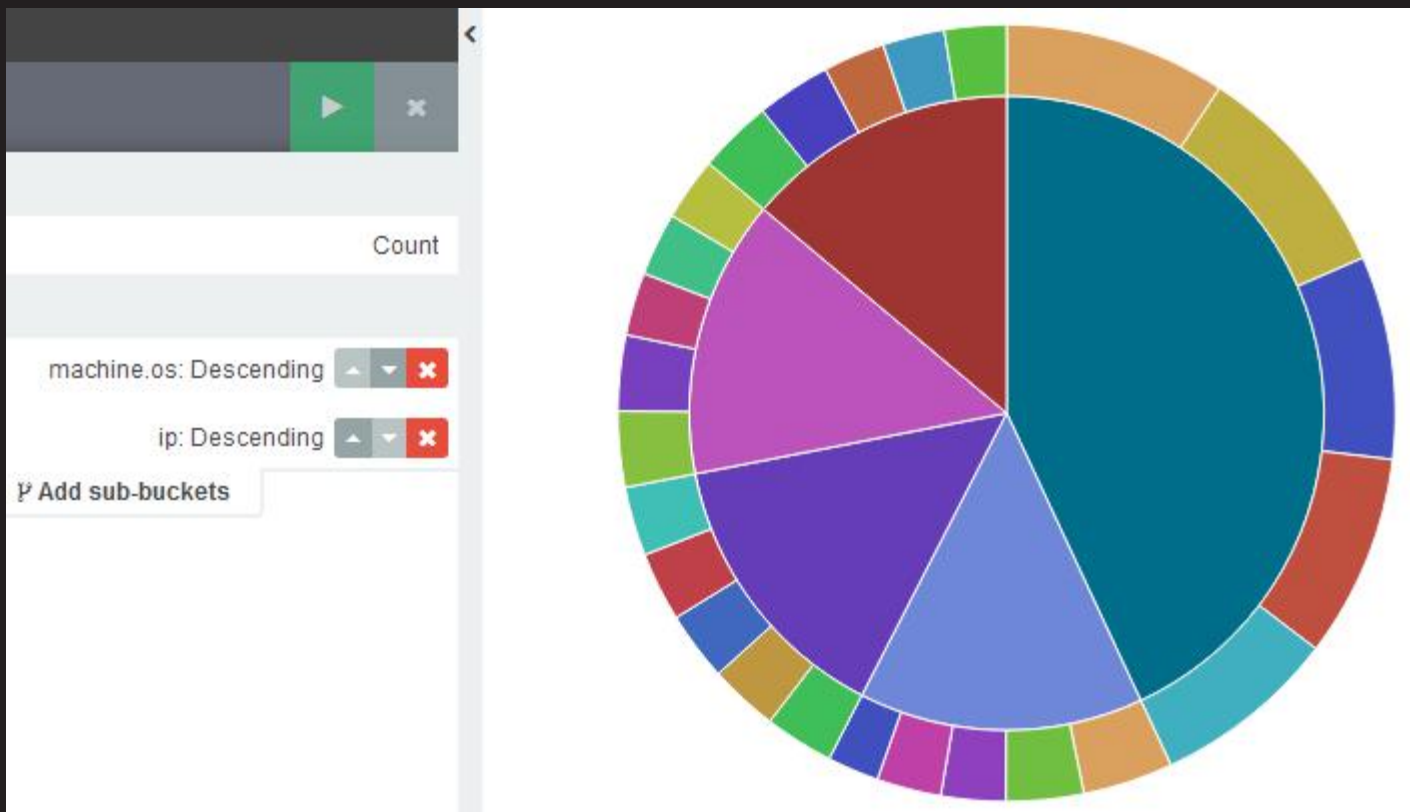
kibana 部分

- 饼图与列表，多种维度自定义统计分析



kibana 部分

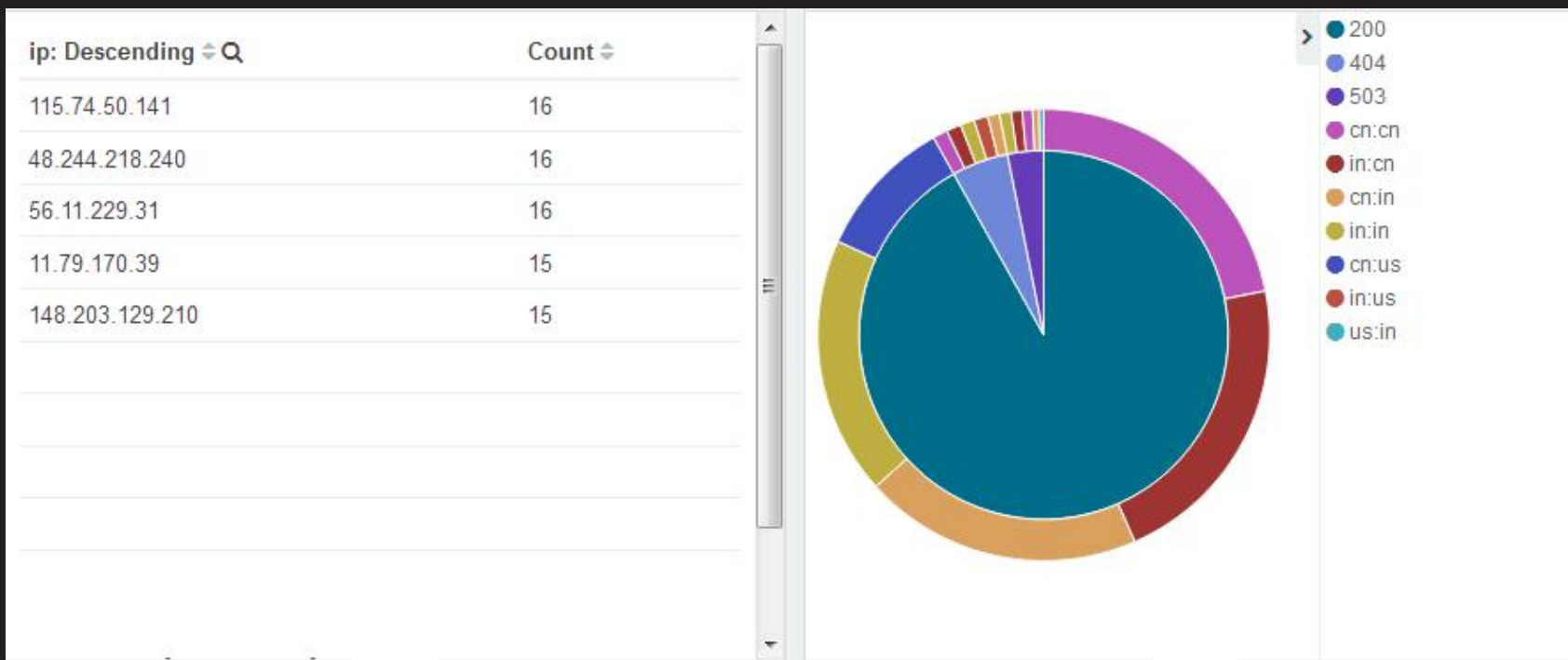
- 饼图与列表，多种维度自定义统计分析



kibana 部分

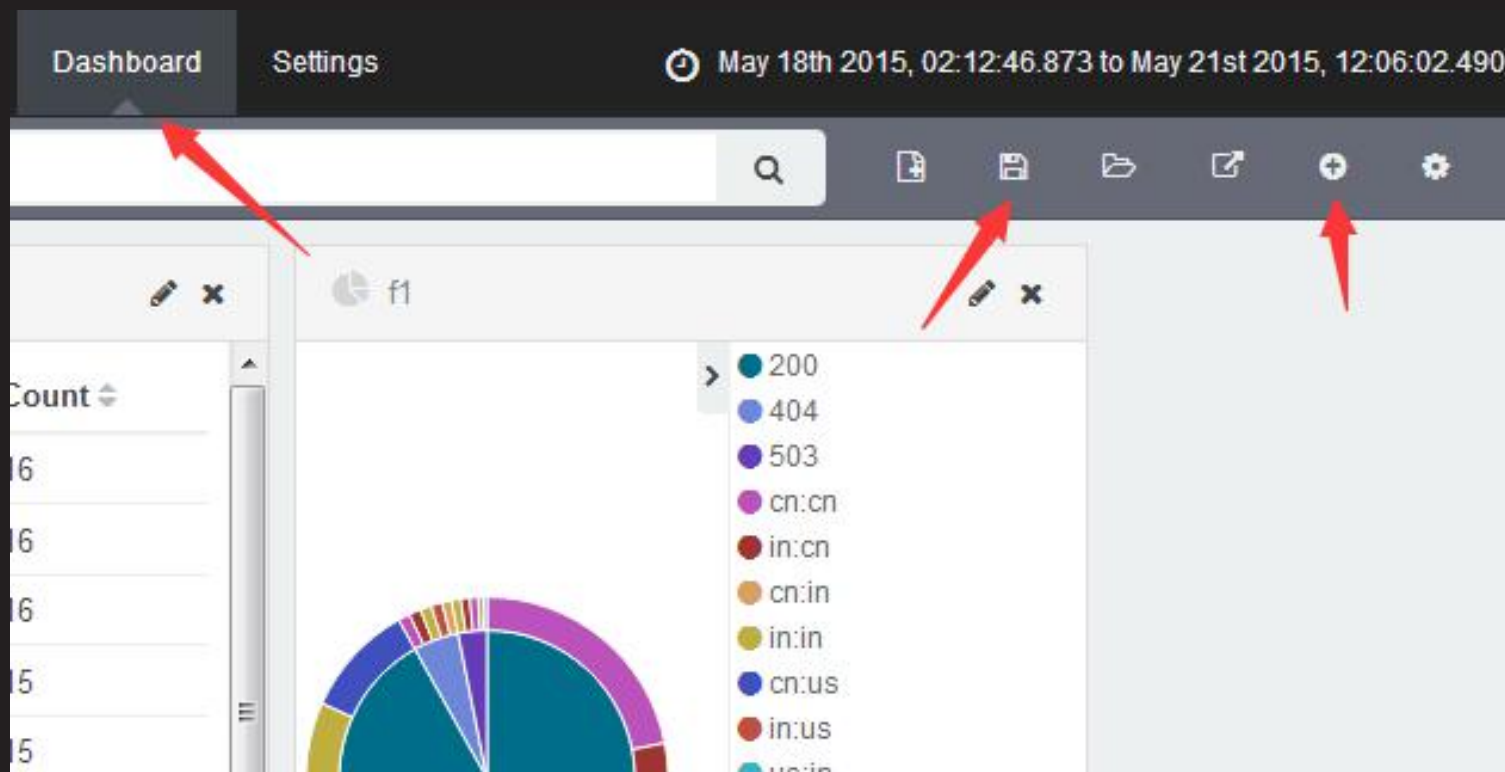
• 列表与饼图

课堂练习



kibana 部分

- 保存后可以在 Dashboard 查看



logstash 部分

logstash 部分

- logstash 是什么
 - logstash是一个数据采集、加工处理以及传输的工具
- logstash 特点：
 - 所有类型的数据集中处理
 - 不同模式和格式数据的正常化
 - 自定义日志格式的迅速扩展
 - 为自定义数据源轻松添加插件



logstash 部分

- logstash 安装
 - Logstash 依赖 java 环境，需要安装 java-1.8.0-openjdk
 - Logstash 没有默认的配置文件，需要手动配置
 - logstash 安装在 /opt/logstash 目录下

```
rpm -ivh logstash-2.3.4-1.noarch.rpm
```



logstash 部分

- logstash 工作结构

- { 数据源 } ==>
- input { } ==>
- filter { } ==>
- output { } ==>
- { ES }



logstash 部分

- logstash 里面的类型
 - 布尔值类型: `ssl_enable => true`
 - 字节类型: `bytes => "1MiB"`
 - 字符串类型: `name => "xkops"`
 - 数值类型: `port => 22`
 - 数组: `match => ["datetime","UNIX"]`
 - 哈希: `options => {k => "v",k2 => "v2"}`
 - 编码解码: `codec => "json"`
 - 路径: `file_path => "/tmp/filename"`
 - 注释: `#`



logstash 部分

- logstash 条件判断

- 等于: ==

- 不等于: !=

- 小于: <

- 大于: >

- 小于等于: <=

- 大于等于: >=

- 匹配正则: =~

- 不匹配正则: !~



logstash 部分

- logstash 条件判断
 - 包含: in
 - 不包含: not in
 - 与: and
 - 或: or
 - 非与: nand
 - 非或: xor
 - 复合表达式: ()
 - 取反符合: !()



logstash 部分

- logstash 的第一个配置文件
 - /etc/logstash/logstash.conf

```
input{  
  stdin}  
}  
filter{ }  
output{  
  stdout}  
}
```

- 启动并验证
 - logstash -f logstash.conf



logstash 部分

- logstash 插件
 - 上页的配置文件使用了 logstash-input-stdin 和 logstash-output-stdout 两个插件，logstash 还有 filter 和 codec 类插件，查看插件的方式是
`/opt/logstash/bin/logstash-plugin list`
 - [插件及文档地址](#)
- 练习
 - logstash 配置从标准输入读取输入源，然后从标准输出输出到屏幕



logstash 部分

- codec类插件
 - 常用的插件：plain、json、json_lines、rubydebug、multiline等
 - 我们还使用刚刚的例子，不过这次我们输入 json 数据
 - 我们设置输入源的 codec 是 json，在输入的时候选择 rubydebug



logstash 部分

- codec 类插件

```
input{
  stdin{ codec => "json" }
}
filter{ }
output{
  stdout{ codec => "rubydebug" }
}
```

- 我们输入普通数据和 json 对比
- {"a": 1, "c": 3, "b": 2}

logstash 部分

- codec 类插件
 - 练习 output 和 input 配置
 - 练习 在 input 不指定类型 json 输出结果
 - 练习 在 output 不指定 rubydebug 的输出结果
 - 同时指定以后的输出结果



logstash 部分

- 练习 input file 插件

```
file{  
    start_position => "beginning"  
    sincedb_path => "/var/lib/logstash/sincedb-access"  
    path => [ "/tmp/alog" , "/tmp/blog" ]  
    type => 'filelog'  
}
```

- sincedb_path 记录读取文件的位置
- start_position 配置第一次读取文件从什么地方开始



logstash 部分

- 练习 input tcp 和 udp 插件

```
tcp{
  host => "0.0.0.0"
  port => 8888
  type => "tcplog"
}
udp{
  host => "192.168.4.16"
  port => 9999
  type => "udplog"
}
```



logstash 部分

- tcp & udp 练习
 - 使用 shell 脚本，对 tcp 指定端口发送数据

```
function sendmsg(){
    if (( $# == 4 )) && [ $1 == "tcp" -o $1 == "udp" ];then
        exec 9<>/dev/$1/$2/$3
        echo "$4" >&9
        exec 9<&-
    else
        echo "$0 (tcp|udp) ipaddr port msg"
    fi
}
```



logstash 部分

- tcp & udp 练习

- 发送 tcp 数据

- `sendmsg tcp 192.168.4.10 8888 'tcp msg'`

- 发送 udp 数据

- `sendmsg udp 192.168.4.10 9999 'udp msg'`



logstash 部分

- syslog 插件练习

```
syslog{  
  host => "192.168.4.10"  
  port => 514  
  type => "syslog"  
}
```

- rsyslog.conf 配置向远程发送数据

```
local0.info                @@192.168.4.10:514
```

- 写 syslog , 查看状态

```
logger -p local0.info -t test_logstash 'test message'
```



logstash 部分

- filter grok插件
 - 解析各种非结构化的日志数据插件
 - grok 使用正则表达式把非结构化的数据结构化
 - 在分组匹配，正则表达式需要根据具体数据结构编写
 - 虽然编写困难，但适用性极广
 - 几乎可以应用于各类数据

```
grok{
    match => [ "message" , "%{IP:ip} , (?<key>reg)" ]
}
```



logstash 部分

- grok 正则分组匹配

- 匹配 ip 时间戳 和 请求方法

```
"(?<ip>(\d+\.){3}\d+) \S+ \S+
(?<time>.*\])\s+\\"(?<method>[A-Z]+)"
```

- 使用正则宏

```
%{IPORHOST:clientip} %{HTTPDUSER:ident} %{USER:auth}
\[%{HTTPDATE:timestamp}\] \"%{WORD:verb}
```

- 最终版本

```
%{COMMONAPACHELOG} \"(?<referer>[^\"]+)\\"
\"(?<UA>[^\"]+)\\"
```



logstash 部分

- input redis 插件

```
redis{
  host => 'redis-server'
  port => '6379'
  data_type => 'list'
  key => 'lb'
  codec => 'json'
}
```

- 生产环境往往理由 redis 来做缓冲，这里给出配置

logstash 部分

- output ES 插件

```
if [type] == "filelog"{
  elasticsearch {
    hosts => ["192.168.4.15:9200"]
    index => "weblog"
    flush_size => 2000
    idle_flush_time => 10
  }
}
```

– 调试成功后，把数据写入 ES 集群

logstash 部分

- input filebeats 插件

```
beats {  
  port => 5044  
  codec => "json"  
}
```

- 这个插件主要用来接收 beats 类软件发送过来的数据，由于 logstash 依赖 java 环境，而且占用资源非常大，我们往往不希望所有集群的机器都部署 java 环境安装 logstash，而使用更轻量的 filebeat 替代



logstash 部分

- filebeat 安装与配置
 - 使用 rpm 安装 filebeat

```
rpm -ivh filebeat-1.2.3-x86_64.rpm
```
 - 修改配置文件 /etc/filebeat/filebeat.yml
 - 设置开机运行

```
systemctl enable filebeat
```
 - 开启服务

```
systemctl start filebeat
```



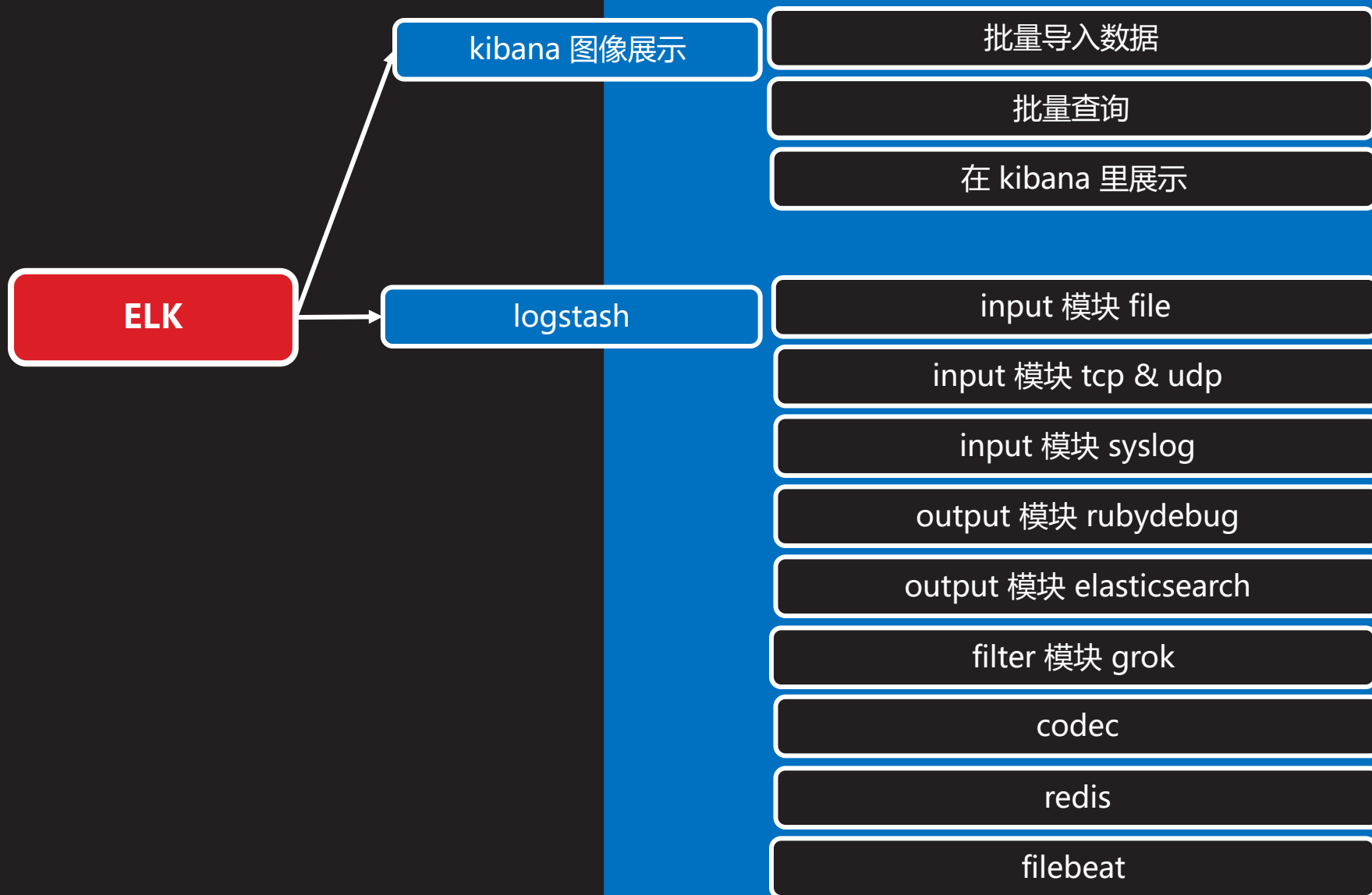
logstash 部分

- 修改配置文件 /etc/filebeat/filebeat.yml

```
paths:
  - /root/logs.jsonl
  document_type: weblog
... ..
paths:
  - /root/accounts.json
  document_type: account
output:
  logstash:
    hosts: ["192.168.4.10:5044"]
```



知识点总结



总结和答疑
