【ELK 技术栈第三天 Logstash Kibana】

主要内容

1. Logstash 简介
2. 安装 Logstash
3. 使用 Logback 向 Logstash 中输出日志
4. Kibana 中操作日志
5. 搭建日志系统

学习目标

知识点	要求
Logstash 简介	掌握
安装 Logstash	掌握
使用 logback 向 Logstash 中输出日志	掌握
Kibana 中操作日志	掌握
搭建日志系统	掌握

一、 LogStash 简介

1 什么是 LogStash

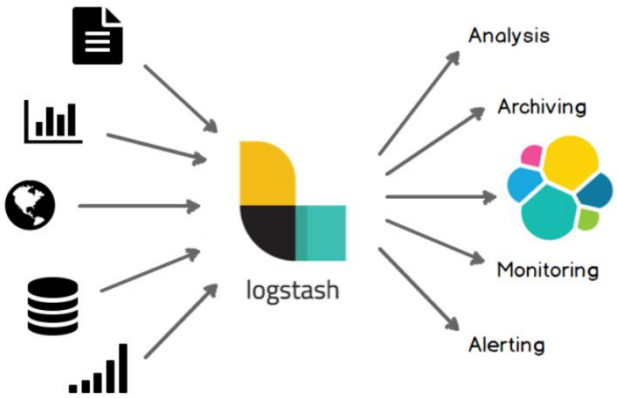
ELK(Elasticsearch+Logstash+Kibana)中我们使用过 Elasticsearch 和 Kibana，就剩下最后一个 LogStash 了。

到底 Logstash 是什么呢？官方说明

官方文字说明：Logstash 是开源的服务器端数据处理管道，能够同时从多个来源采集数据，转换数据，然后将数据发送到您最喜欢的“存储库”中。

通俗说明：Logstash 是一款强大的数据处理工具，常用作日志处理。

到目前为止，Logstash 已经有超过 200 个可用的插件，以及创建和贡献自己的灵活性。社区生态非常完善，对于我们可以放心的使用。



2 为什么使用 Logstash

通常当系统发生故障时，工程师需要登录到各个服务器上，使用 grep / sed / awk 等 Linux 脚本工具去日志里查找故障原因。在没有日志系统的情况下，首先需要定位处理请求的服务器，如果这台服务器部署了多个实例，则需要去每个应用实例的日志目录下去找日志

文件。每个应用实例还会设置日志滚动策略（如：每天生成一个文件），还有日志压缩归档策略等。

这样一系列流程下来，对于我们排查故障以及及时找到故障原因，造成了比较大的麻烦。因此，如果我们能**把这些日志集中管理**，并提供集中检索功能，不仅可以提高诊断的效率，同时对系统情况有个全面的理解，避免事后救火的被动。

所以日志集中管理功能就可以使用 ELK 技术栈进行实现。Elasticsearch 只有数据存储和分析的能力，Kibana 就是可视化管理平台。还缺少数据收集和整理的角色，这个功能就是 Logstash 负责的。

3 Logstash 工作原理

3.1 Data Source

Logstash 支持的数据源有很多。例如对于日志功能来说只能能有日志记录和日志传递功能的日志都支持，Spring Boot 中默认推荐 logback 支持日志输出功能（输出到数据库、数据出到文件）。

我们就使用 logback 进行日志输出给 Logstash。

3.2 Logstash Pipeline

整个整体就是 Logstash 的功能。

在 Logstash 中包含非常重要的三个功能：

a) Input

输入源，一般配置为自己监听的主机及端口。DataSource 向指定的 ip 及端口输出日志，Input 输入源监听到数据信息就可以进行收集。

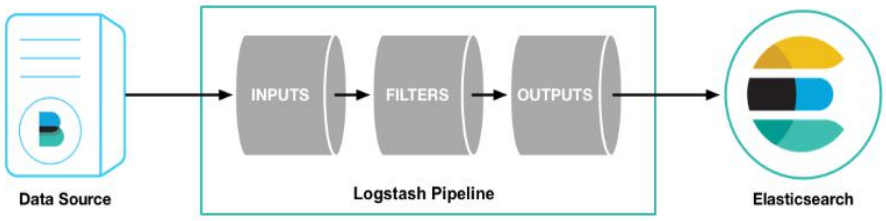
b) Filter

过滤功能，对收集到的信息进行过滤（额外处理），也可以省略这个配置（不做处理）

c) Output

把收集到的信息发送给谁。在 ELK 技术栈中都是输出给 Elasticsearch，后面数据检索和数据分析的过程就给 Elasticsearch 了。

最终效果：通过整体步骤就可以把原来一行日志信息转换为 Elasticsearch 支持的 Document 形式（键值对形式）的数据进行存储。



二、 安装 Logstash

在前面的课程中已经安装好了 Elasticsearch 和 Kibana。下面是安装 Logstash 的步骤

Logstash 是不需要必须和 Elasticsearch 安装到一起，如果独立安装到一台服务器，需要在服务器中先配置好 JDK 环境变量。在课堂中把 ELK 三个软件都装到一台服务器中。

1 安装 Logstash

1.1 上传 Logstash 并解压

上传压缩包到/usr/local/tmp 中后，解压压缩包。

tar zxf logstash-6.8.4.tar.gz

剪切到/usr/local 中并命名为 logstash

mv logstash-6.8.4 ../logstash

1.2 修改配置

进入到 logstash 配置文件夹中

cd /usr/local/logstash/config/

创建配置文件，名称自定义。

vim mylogstash.conf

配置解释说明：

input:接收日志输入配置

tcp: 协议

mode: logstash 服务

host:logstash 主机 ip

port：端口，自己指定。默认 4560

output：日志处理输出

elasticsearch: 交给 es 处理

action：es 中 index 命令。也就是新增命令。

hosts：es 的主机

index:存储日志的索引。如果不存在可以自动创建。默认的 type 名称为 doc

一定要先启动编辑状态（点击键盘 i 键）在粘贴，如果没启用第一行是 nput{少个 i。

```
input {
        tcp {
                mode => "server"
                host => "192.168.8.140"
                port => 4560
        }
}
filter {
}
output {
```

```
        elasticsearch {
                action => "index"
                hosts    => "192.168.8.140:9200"
                index    => "test_log"
        }
}
```

1.3 启动 Logstash

进入到 bin 目录

cd /usr/local/logstash/bin

需要先启动 Elasticsearch 否则会频繁提示无法连接到 Elasticsearch

```
[2020-04-10T21:47:09,737][WARN ][logstash.outputs.elasticsearch] Attempted to resurrect connection to dead ES instance, but got an error. {:
url=>"http://192.168.8.140:9200/", :error_type=>LogStash::Outputs::ElasticSearch::HttpClient::Pool::HostUnreachableError, :error=>"Elasticse
arch Unreachable: [http://192.168.8.140:9200/][Manticore::SocketException] Connection refused (Connection refused)"}
```

启动 logstash

./logstash -f /usr/local/logstash/config/mylogstash.conf

如果启动完成没有出异常，提示 Successfully 说明安装成功。

```
[2020-04-10T10:36:39,236][INFO ][logstash.outputs.elasticsearch] Using default mapping template
[2020-04-10T10:36:39,281][INFO ][logstash.outputs.elasticsearch] Attempting to install template {:manage_template=>{"template"=>"logstash-*"
, "version"=>60001, "settings"=>{"index.refresh_interval"=>"5s"}, "mappings"=>{"_default_"=>{"dynamic_templates"=>[{"message_field"=>{"path_
match"=>"message", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false}}}, {"string_fields"=>{"match"=>"*", "match_ma
pping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false, "fields"=>{"keyword"=>{"type"=>"keyword", "ignore_above"=>256}}}}}], "pro
perties"=>{"@timestamp"=>{"type"=>"date"}, "@version"=>{"type"=>"keyword"}, "geoip"=>{"dynamic"=>true, "properties"=>{"ip"=>{"type"=>"ip"},
"location"=>{"type"=>"geo_point"}, "latitude"=>{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}}}}
[2020-04-10T10:36:39,968][INFO ][logstash.pipeline        ] Pipeline started successfully {:pipeline_id=>"main", :thread=>"#<Thread:0x58f0c0
63 run>"}
[2020-04-10T10:36:39,982][INFO ][logstash.inputs.tcp      ] Starting tcp input listener {:address=>"192.168.8.140:4560", :ssl_enable=>"false
"}
[2020-04-10T10:36:40,180][INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :non_running_pipeline
s=>[]}
[2020-04-10T10:36:40,736][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
```

三、 使用 Logback 向 Logstash 中输出日志

需求：随意新建一个项目把输出到控制台的日志信息也输出到 Logstash 中。

1 修改 pom.xml

logstash-logback-encoder 就是转码后向 logstash 中输入的依赖。

注意：

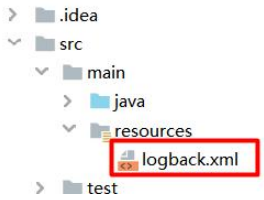
如果导入的是 6.x 版本不会在控制台看见任何额外日志信息。

如果导入的是 5.x 版本会在控制台看见 logback.xml 加载的信息。

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.6.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>net.logstash.logback</groupId>
        <artifactId>logstash-logback-encoder</artifactId>
        <version>6.3</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

2 导入 logback.xml

在当天授课目录/软件 中有 logback.xml。把文件粘贴到 resources 中。



logback.xml 文件内容如下，红色部分表示向 logstash 中输出日志信息。

红色中<destination>配置的是 logstash 配置文件中 input 里面 host 和 post 的信息。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--该日志将日志级别不同的 log 信息保存到不同的文件中 -->
<configuration>
   <include
resource="org/springframework/boot/logging/logback/defaults.xml" />

   <springProperty scope="context" name="springAppName"
      source="spring.application.name" />
```

```
    <!-- 日志在工程中的输出位置 -->
    <property name="LOG_FILE"
value="${BUILD_FOLDER:-build}/${springAppName}" />

    <!-- 控制台的日志输出样式 -->
    <property name="CONSOLE_LOG_PATTERN"
        value="%clr(%d{yyyy-MM-dd
HH:mm:ss.SSS}){faint} %clr(${LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:- }){mag
enta} %clr(---){faint} %clr([%15.15t]){faint} %m%n${LOG_EXCEPTION_CONVERS
ION_WORD:-%wEx}}" />

    <!-- 控制台输出 -->
    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>INFO</level>
        </filter>
        <!-- 日志输出编码 -->
        <encoder>
            <pattern>${CONSOLE_LOG_PATTERN}</pattern>
            <charset>utf8</charset>
        </encoder>
    </appender>
     <!-- logstash 远程日志配置-->
     <appender name="logstash"
class="net.logstash.logback.appender.LogstashTcpSocketAppender">
         <destination>192.168.8.140:4560</destination>
         <encoder charset="UTF-8"
class="net.logstash.logback.encoder.LogstashEncoder" />
     </appender>
    <!-- 日志输出级别 -->
    <root level="DEBUG">
        <appender-ref ref="console" />
        <appender-ref ref="logstash" />
    </root>
</configuration>
```

3 新建启动类

新建 com.bjsxt.DemoApplication

```
@SpringBootApplication
```

```
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class,args);
    }
}
```

四、 在 Kibana 中查看日志信息

1 使用命令方式查看

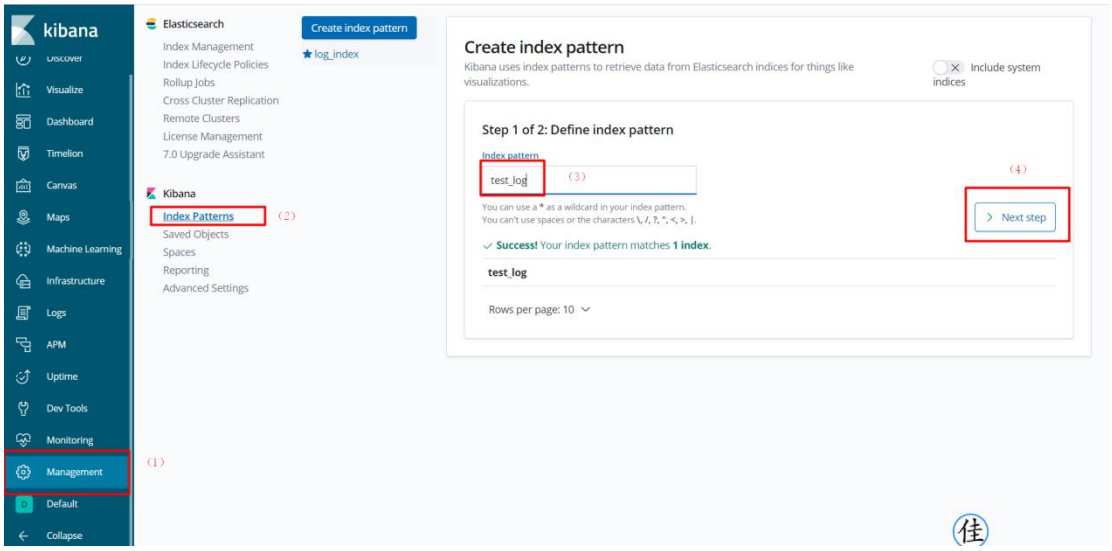
可以直接在 Dev Tools 中输入命令查看日志信息。

输入： GET test_log/_search 查看全部。

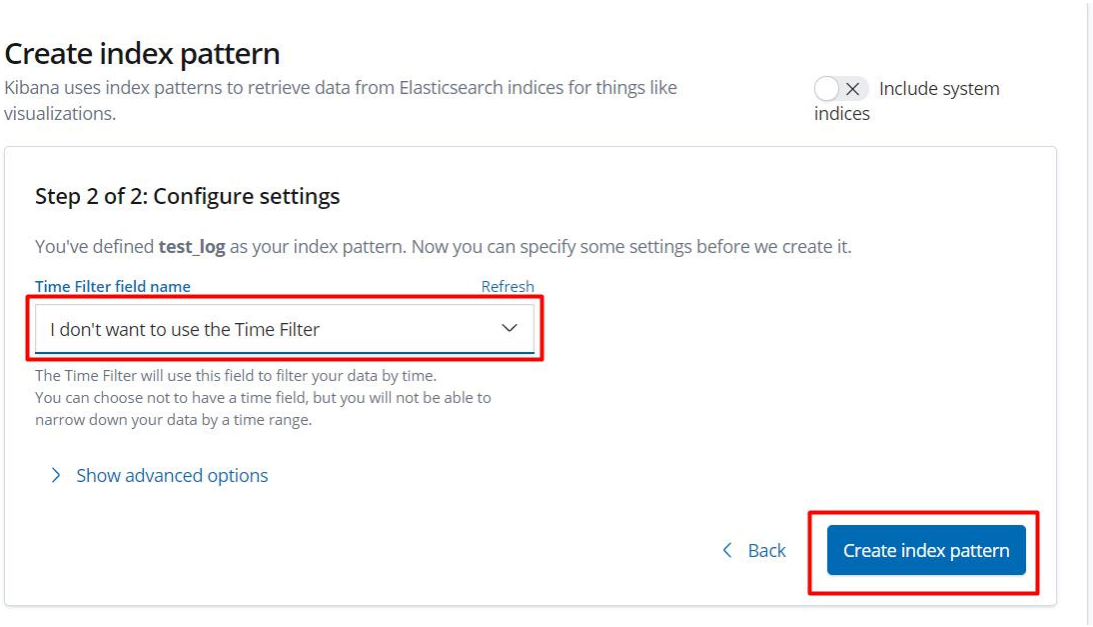

```
{
  "took" : 11,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 10,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "test_log",
        "_type" : "doc",
        "_id" : "Lu0PZ3EBXvcA7lFerNiP",
        "_score" : 1.0,
        "_source" : {
          "@timestamp" : "2020-04-11T02:26:52.182Z",
          "port" : 53692,
          "@version" : "1",
          "host" : "192.168.8.1",
          "message" : """{"@timestamp":"2020-04-11T10:26:52.283+08:00","@version":"1","message":"No active profile
            set, falling back to default profiles: default","logger_name":"com.bjsxt.DemoApplication","thread_name"
            :"main","level":"INFO","level_value":20000}
"""
        }
      },
```


2 是 Kibana 界面查看

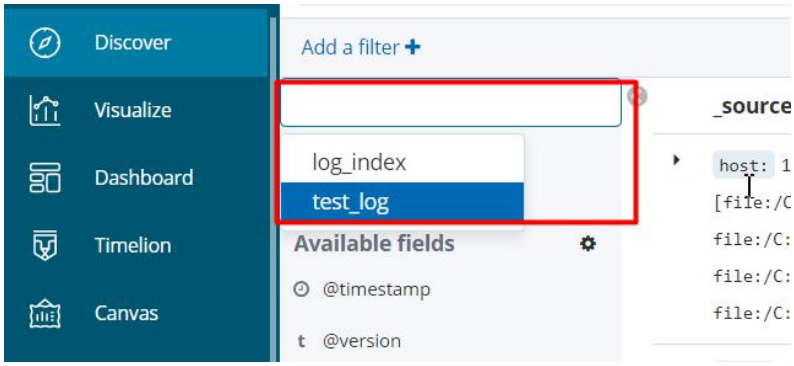
进入到 Kibana 后按图所示点击。创建索引表达式



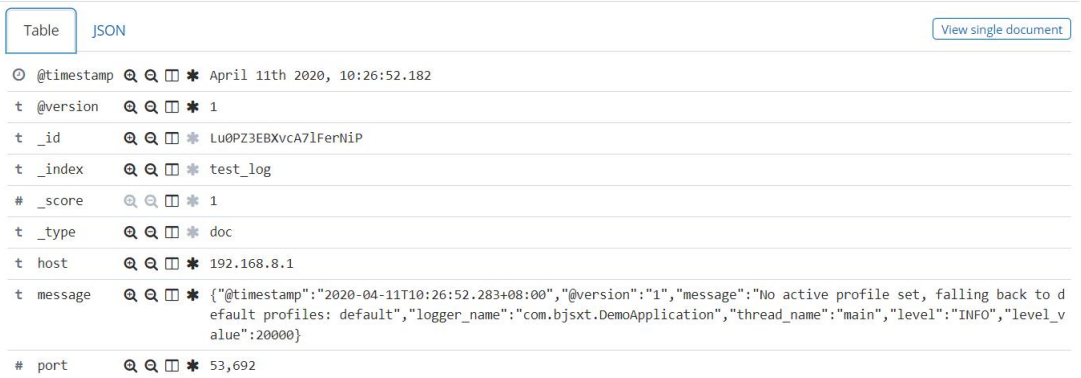
选择没有时间过滤后，点击“Create index pattern”按钮



点击菜单中 Discover，选择右侧 test_log



每条日志在 Elasticsearch 中存储形式



IDEA 中控制台打印的原日志内容是下面内容。Logstash 作用就是把下面内容转换为上面 Elasticsearch 存储的内容。在中间做了数据格式转换，收集数据放入 Elasticsearch 中的工作。

```
INFO 8304 --- [           main] No active profile set, falling back to default profiles: default
```

五、 搭建日志系统

绝大多数项目在后台管理中都有日志管理。以前的日志信息是存储在 MySQL 中，日志随着项目运行时间会越来越多，一直存储在 MySQL 会导致查询降低。现在的日志信息通过 ELK 技术栈进行操作。存储在 Elasticsearch 中，可以更好的分析日志内容及更快查询效率。

给定简单需求：

搭建日志系统，提供查询 Elasticsearch 中日志信息的接口。

1 新建项目

名称为 ELK_Demo

2 修改 pom.xml

搭建最基本的环境，实现需求，没有考虑 Spring Cloud 相关环境，如果考虑 Spring Cloud 还需要配置 Eureka 等信息。

```
<parent>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.6.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.12</version>
    </dependency>
</dependencies>
```

3 创建配置文件

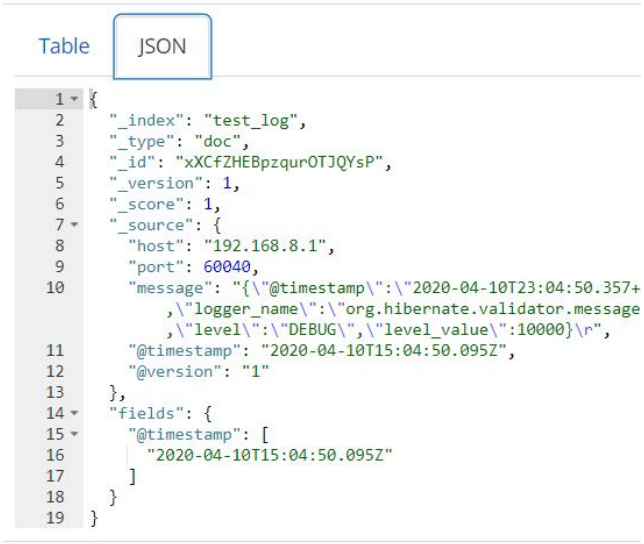
在 resources 下新建 application.yml 配置文件。

配置 Elasticsearch 相关配置信息。

```
spring:
  data:
    elasticsearch:
      cluster-name: elasticsearch
      cluster-nodes: 192.168.8.140:9300
```

4 新建实体

根据 kibana 中查看到日志信息可以得出看出，除了 message 是类类型，里面包含一些其他属性外，其他的属性都是简单类型属性。



新建 com.bjsxt.pojo.Log。

注意@version 和@timestamp 要使用@JsonProperty 进行接收。

```
@Data
@Document(indexName = "test_log",type = "doc")
public class Log {
    @Id
    private String id;
    @Field(type= FieldType.Text)
    private String host;
    @Field(type= FieldType.Text)
    private String message;
    @Field(type= FieldType.Long)
    private Long port;
    @Field(type = FieldType.Date)
    @JsonProperty("@timestamp")
    private Date timestamp;
    @Field(type = FieldType.Text)
    @JsonProperty("@version")
    private String version;
}
```

5 新建 service 及实现类

新建 com.bjsxt.service.LogService 及实现类

```
public interface LogService {
```

```
    List<Log> selectByPage(int page,int size);
}
```

```
@Service
public class LogServiceImpl implements LogService {
    @Autowired
    private ElasticsearchTemplate elasticsearchTemplate;
    @Override
    public List<Log> selectByPage(int page, int size) {
        SearchQuery sq = new
NativeSearchQuery(QueryBuilders.matchAllQuery());
        sq.setPageable(PageRequest.of(page-1,size));
        return elasticsearchTemplate.queryForList(sq,Log.class);
    }
}
```

6 新建控制器

新建 com.bjsxt.controller.LogController

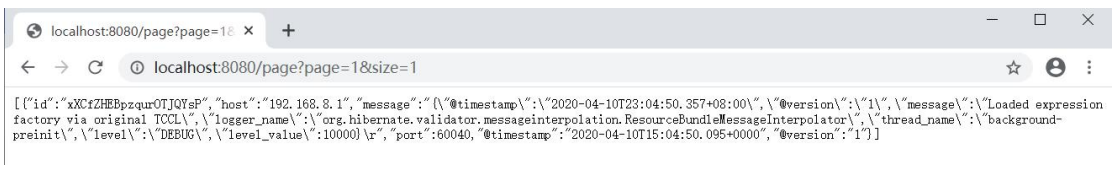
```
@Controller
public class LogController {
    @Autowired
    private LogService logService;

    @RequestMapping("/page")
    @ResponseBody
    public List<Log> showPage(int page,int size){
        return logService.selectByPage(page,size);
    }
}
```

7 测试结果

在浏览器输入: http://localhost:8080/page?page=1&size=2

会看见下面的结果。



六、 LogStash+MySQL+Elasticsearch 实现数据增量导入（双写一致）

原有系统中，如果使用了缓存应用，全文搜索服务等额外数据存储，则在代码实现中，要保证双写一致，即写数据库的同时，把数据的变量同步到其他存储中。

如果使用 LogStash，则可以实现数据的增量导入。

思路：写数据到数据库，LogStash 监听数据库中数据的变化，把增量数据读取，并保存到 ES 中。

1 环境准备

1.1 上传数据库驱动

LogStash 本身不提供数据库驱动，需要使用者提供数据库的驱动包，且 LogStash 中的数据库 JDBC 插件就是 Java 开发的。需要上传数据库驱动到 LogStash 所在主机。

Logstash5.x & 6.3.*以下版本，上传驱动不需要固定位置，任意位置即可。

Logstash6.8.4 版本的上传位置固定是：$LogStash_HOME/logstash_core/lib/jars/

1.2 准备数据库表格

案例中使用电商项目中的商品表格，建表语句如下：

```
CREATE TABLE `tb_item` (
  `id` bigint(20) NOT NULL COMMENT '商品 id，同时也是商品编号',
  `title` varchar(100) NOT NULL COMMENT '商品标题',
  `sell_point` varchar(500) DEFAULT NULL COMMENT '商品卖点',
  `price` bigint(20) NOT NULL COMMENT '商品价格，单位为：分',
```

```
    `num` int(10) NOT NULL COMMENT '库存数量',
    `barcode` varchar(30) DEFAULT NULL COMMENT '商品条形码',
    `image` varchar(500) DEFAULT NULL COMMENT '商品图片',
    `cid` bigint(10) NOT NULL COMMENT '所属类目，叶子类目',
    `status` tinyint(4) NOT NULL DEFAULT '1' COMMENT '商品状态，1-正常，
2-下架，3-删除',
    `created` datetime NOT NULL COMMENT '创建时间',
    `updated` datetime NOT NULL COMMENT '更新时间',
    PRIMARY KEY (`id`),
    KEY `cid` (`cid`),
    KEY `status` (`status`),
    KEY `updated` (`updated`)
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='商品表';
```

LogStash 实现增量导入，需要有一个定位字段，这个字段的数据，可以表示数据的新旧，代表这个数据是否是一个需要导入到 ES 中的数据。案例中使用表格的 updated 字段作为定位字段，每次读取数据的时候，都会记录一个最大的 updated 时间，每次读取数据的时候，都读取 updated 大于等于记录的定位字段数据。每次查询的就都是最新的，要导入到 ES 中的数据。

2 编写 LogStash 配置文件

在$LogStash_home/config/目录中，编写配置文件 ego-items-db2es.conf

vim config/ego-items-db2es.conf

```
input {
  jdbc {
    # 连接地址
```

```
        jdbc_connection_string                                          =>
"jdbc:mysql://192.168.1.2:3306/ego?useUnicode=true&characterEncoding=UT
F-8&serverTimezone=UTC"

        # 数据库用户名和密码
        jdbc_user => "root"
        jdbc_password => "root"

        # 驱动类，如果使用低版本的 logstash，需要再增加配置
jdbc_driver_library，配置驱动包所在位置
        jdbc_driver_class => "com.mysql.cj.jdbc.Driver"
        # 是否开启分页逻辑
        jdbc_paging_enabled => true
        # 分页的长度是多少
        jdbc_page_size => "2000"
        # 时区
        jdbc_default_timezone => "Asia/Shanghai"

        # 执行的 SQL
        statement => "select id, title, sell_point, price, image, updated
from tb_item where updated >= :sql_last_value order by updated asc"

        # 执行 SQL 的周期，[秒] 分钟 小时 天 月 年
        schedule => "* * * * *"

        # 是否使用字段的值作为比较策略
        use_column_value => true
        # 作为比较策略的字段名称
        tracking_column => "updated"
        # 作为比较策略的字段类型，可选为 numberic 和 timestamp
        tracking_column_type => "timestamp"
```

```
        # 记录最近的比较策略字段值的文件是什么 ,相对寻址路径是 logstash 的安装路
径
        last_run_metadata_path => "./ego-items-db2es-last-value"
        # 是否每次执行 SQL 的时候，都删除 last_run_metadata_path 文件内容
        clean_run => false
        # 是否强制把 ES 中的字段名都定义为小写。
        lowercase_column_names => false
      }
    }

    output {
      elasticsearch {
        hosts               =>              ["http://192.168.89.140:9200",
"http://192.168.89.141:9200"]
        index => "ego-items-index"
        action => "index"
        document_id => "%{id}"
      }
    }
```

3 安装 logstash-input-jdbc 插件

在 LogStash6.3.x 和 5.x 版本中，logstash-input-jdbc 插件是默认安装的。在 6.8.4 版本的 LogStash 中是未安装的，需要手工安装。安装命令如下：

$Logstash_HOME/bin/logstash-plugin install logstash-input-jdbc

4 启动测试

启动 LogStash 命令不变：

bin/logstash -f 配置文件