

日志管理

- 1 KafkaAppender使用
 - 1.1 配置文件
- 2 修改logback-spring.xml
 - 2.1 配置Kibana
 - 2.1.1 配置日志的数据源
 - 2.1.2 修改系统配置（视情况可选配置）
 - 2.1.3 查看日志
- 3 文件方式管理日志
- 4 指定logger的配置
- 5 代码中日志使用
 - 5.1 lombok方式
 - 5.2 debug级别日志：

通用规则

代码开发中，本地日志只打印控制台。

服务器日志可以通过admin页面查看在线日志。在线日志循环输出，默认保留10M大小。

服务器日志使用DailyFileAppender与KafkaAppender。

在容器平台部署的服务，推荐使用kafkaAppender，日志最终会保存在elasticsearch中，通过kibana提供日志的查询。虚拟机部署的应用可以使用DailyFileAppender

KafkaAppender使用

配置文件

使用kafkaAppender

修改logback-spring.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <include
        resource="org/springframework/boot/logging/logback/base.
xml" />
    <jmxConfigurator />

    <appender name="kafkaAppender"
        class="araf.kafka.appender.KafkaAppender">
        <encoder>
            <pattern>${CONSOLE_LOG_PATTERN:-%clr(%d{yyyy-MM-dd
                HH:mm:ss.SSS}){faint} %clr
($ {LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:-
                }){magenta} %clr(---){faint} %clr([%15.15
t]){faint}
                %clr(%-40.40logger{39}){cyan} %clr(:)
{faint}
                %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}
            </pattern>
        </encoder>

    </appender>

    <logger name="com.acca.opra">
        <appender-ref ref="kafkaAppender" /> root appender
    </logger>

    <root>
        <appender-ref ref="kafkaAppender" /> kafka
    </root>
</configuration>

```

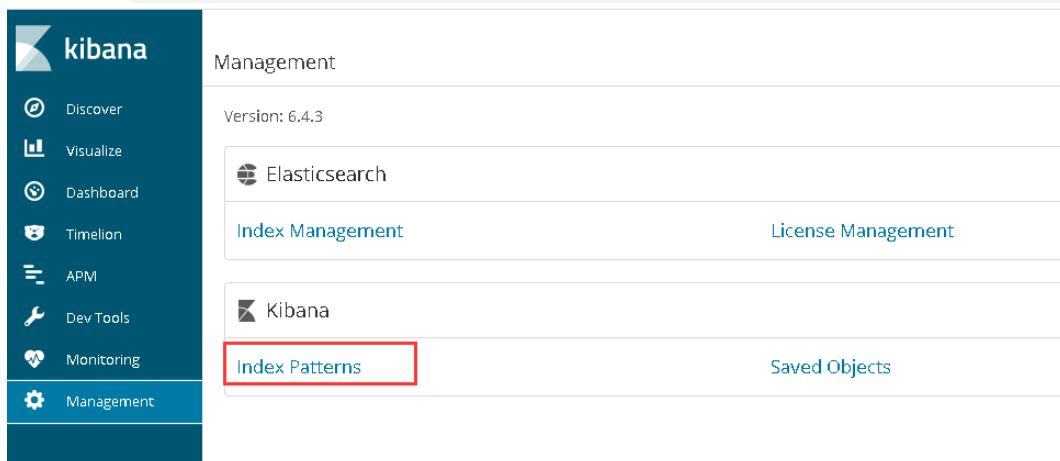
配置Kibana

登录kibana, 开发环境地址: <http://10.1.17.13:5601/>

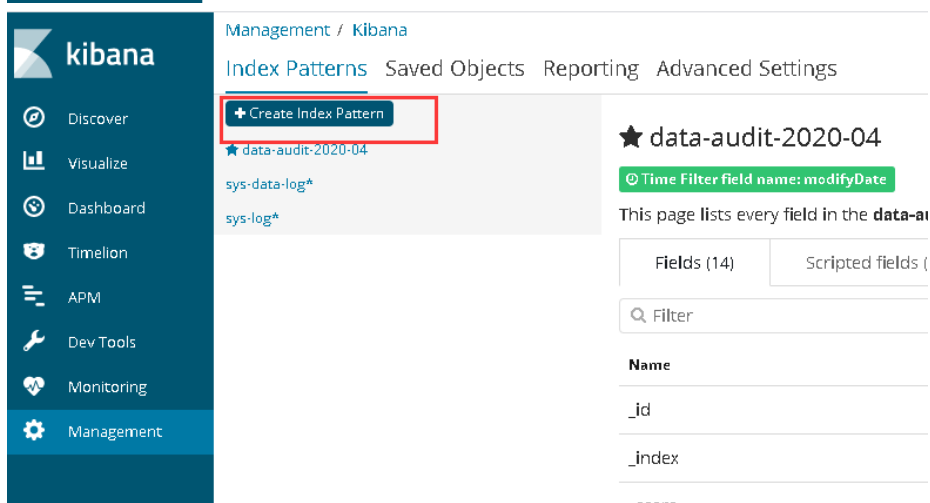
环境	地址
开发	http://10.1.17.13:5601/
测试	http://10.1.19.223:5601/

配置日志的数据源

10.1.17.13:5601/app/kibana#/management?_g=(refreshInterval:(paused:false,value:5000),time:(from:now-15m,mode:quick))



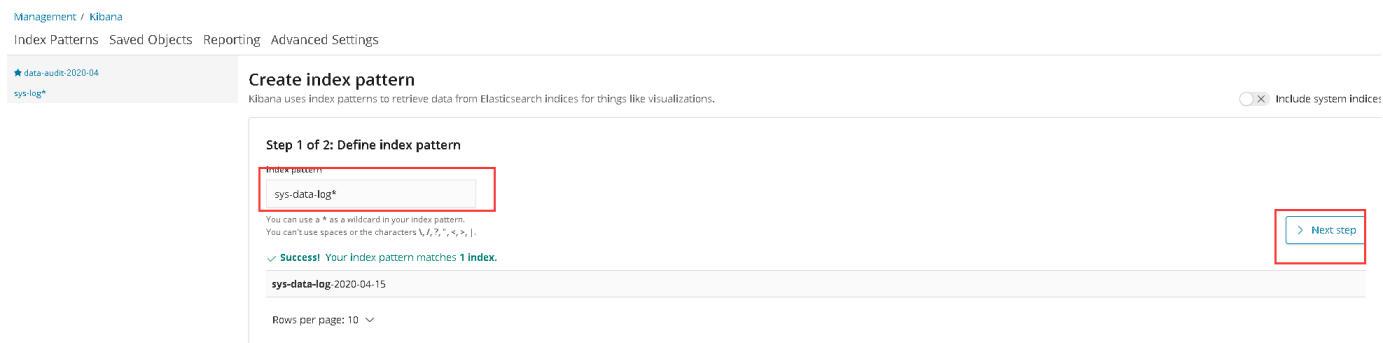
The screenshot shows the Kibana Management page. The left sidebar contains navigation links: Discover, Visualize, Dashboard, Timelion, APM, Dev Tools, Monitoring, and Management (highlighted). The main content area is titled 'Management' and shows 'Version: 6.4.3'. Below this, there are two sections. The first section is for 'Elasticsearch' and contains links for 'Index Management' and 'License Management'. The second section is for 'Kibana' and contains links for 'Index Patterns' (highlighted with a red box) and 'Saved Objects'.



The screenshot shows the Kibana Index Patterns page. The left sidebar is the same as the previous screenshot. The main content area has a breadcrumb 'Management / Kibana' and tabs for 'Index Patterns', 'Saved Objects', 'Reporting', and 'Advanced Settings'. The 'Index Patterns' tab is active, showing a '+ Create Index Pattern' button (highlighted with a red box) and a list of index patterns: 'data-audit-2020-04', 'sys-data-log*', and 'sys-log*'. On the right, there is a section for 'data-audit-2020-04' with a 'Time Filter field name: modifyDate' and a table listing fields. The table has two tabs: 'Fields (14)' and 'Scripted fields (0)'. The 'Fields (14)' tab is active, showing a search filter and a table with columns 'Name' and 'Type'. The table lists fields: '_id', '_index', and 'score'.

indexpattern的格式为: {spring.application.name}-log*

spring.application.name为自己项目中配置的名称



The screenshot shows the Kibana 'Create index pattern' page. The left sidebar is the same as the previous screenshots. The main content area has a breadcrumb 'Management / Kibana' and tabs for 'Index Patterns', 'Saved Objects', 'Reporting', and 'Advanced Settings'. The 'Index Patterns' tab is active, showing a '+ Create Index Pattern' button (highlighted with a red box) and a list of index patterns: 'data-audit-2020-04' and 'sys-log*'. On the right, there is a section for 'data-audit-2020-04' with a 'Time Filter field name: modifyDate' and a table listing fields. The table has two tabs: 'Fields (14)' and 'Scripted fields (0)'. The 'Fields (14)' tab is active, showing a search filter and a table with columns 'Name' and 'Type'. The table lists fields: '_id', '_index', and 'score'. Below this, there is a section for 'Create index pattern' with a text input field for 'Index pattern:' containing 'sys-data-log*' (highlighted with a red box). Below the input field, there is a success message: 'Success! Your index pattern matches 1 index.' and a list of matching index patterns: 'sys-data-log: 2020-04-15'. At the bottom right, there is a 'Next step' button (highlighted with a red box).

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☐ Include system indices

Step 2 of 2: Configure settings

You've defined **sys-data-log*** as your index pattern. Now you can specify some settings before we create it.

Time filter field name

timestamp

The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

[Show advanced options](#)[< Back](#)[Create Index pattern](#)

Time picker defaults

The timefilter selection to use when Kibana is started without one

Default:

```
{
  "from": "now-15m",
  "to": "now",
  "mode": "quick"
}
```

timepicker:timeDefaults

```
{
  "from": "now/d",
  "to": "now/d",
  "display": "Today",
  "section": 0
}
```

[Reset to default](#)

将默认时间显示从15分钟改成Today

```
{
  "from": "now/d",
  "to": "now/d",
  "display": "Today",
  "section": 0
}
```

修改系统配置（视情况可选配置）

elasticsearch中的时间格式默认不带时区，如果显示的时间不正确，可以调整一下kibana的默认时区设置。

在Management-Advance Setting中，设置时区为对应的时区

Timezone for date formatting

Which timezone should be used. "Browser" will use the timezone detected by your browser.

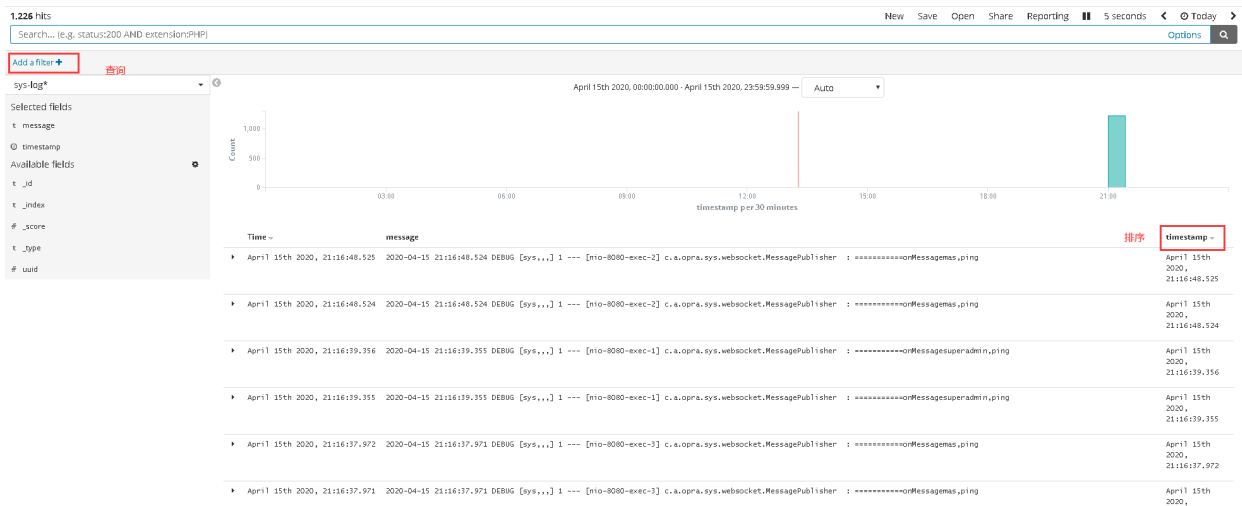
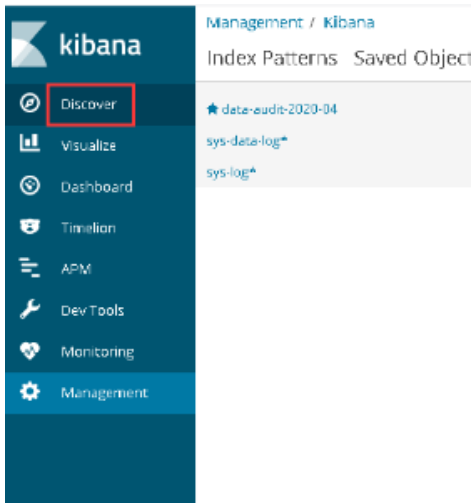
Default: **Browser**

dateFormat:tz

Etc/GMT+0

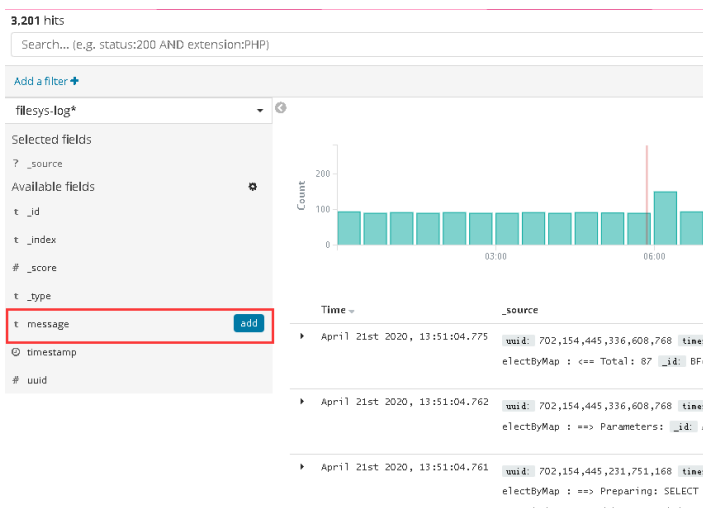
[Reset to default](#)

查看日志

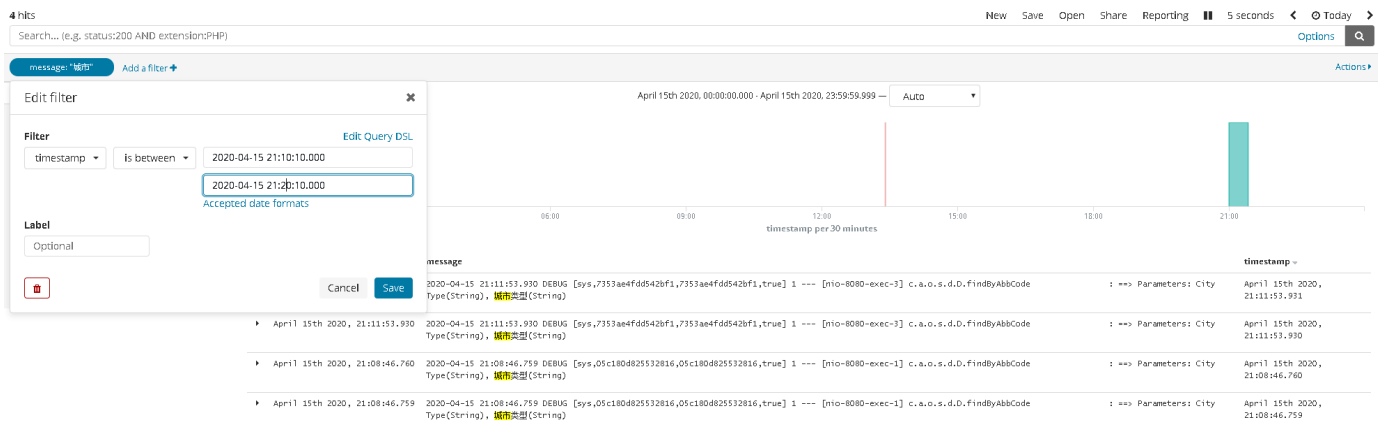
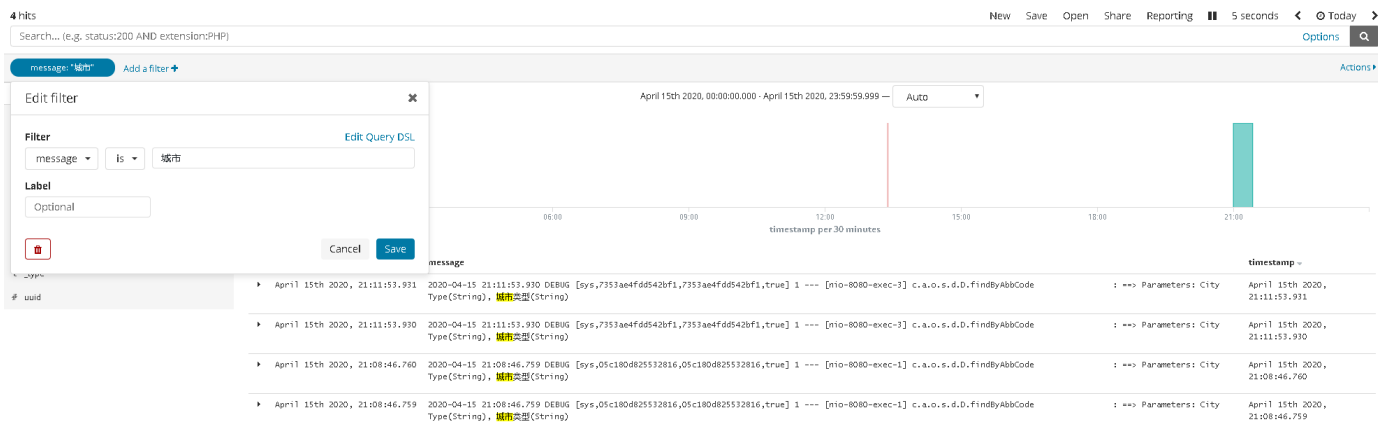


选择indexPattern，选择自动刷新时间。

添加message字段到右侧视图。

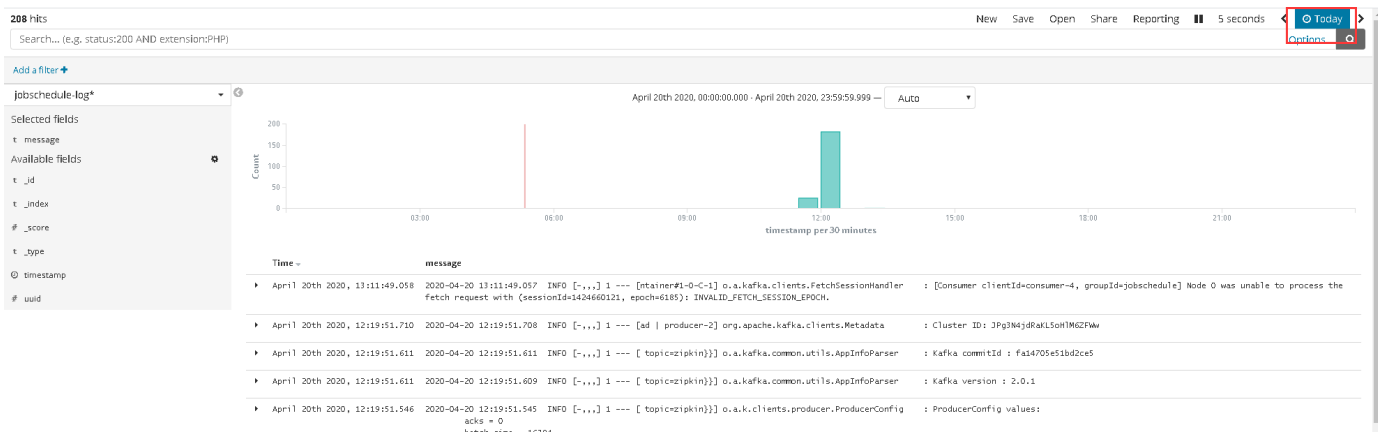


选择add filter 查询日志。可以按时间范围查询，也可以查询日志中的所有数据



注意日期格式必须为: yyyy-MM-dd HH:mm:ss.SSS

点击这里可以切换时间



文件方式管理日志

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include
    resource="org/springframework/boot/logging/logback/base.xml" />
  <jmxConfigurator />

  <property name="fileName" value="@project.artifactId@" />

  <appender name="rollingFile"
    class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>logs/${fileName}.log</file>
    <rollingPolicy
      class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <fileNamePattern>logs/${fileName}/${fileName}.%d{yyyy-MM-
dd}.log
      </fileNamePattern>
    </rollingPolicy>
    <encoder>
      <pattern>${FILE_LOG_PATTERN:-%d{yyyy-MM-dd HH:mm:ss.SSS}
      ${LOG_LEVEL_PATTERN:-%5p} ${PID:- } --- [%t] %-40.40
logger{39} :
      %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}
    </pattern>
    </encoder>
  </appender>

  <root level="INFO">
    <appender-ref ref="rollingFile" />
    <appender-ref ref="kafkaAppender" />
  </root>

</configuration>

```

指定logger的配置

某些场景中，对于特殊的代码位置，可以执行特定的日志输出级别等，可以参考下面的配置。

配置文件中配置

```

<configuration>
    <include
        resource="org/springframework/boot/logging/logback/base.
xml" />
    <jmxConfigurator />

    <property name="fileName" value="@project.artifactId@" />
    <appender name="kafkaAppender"
        class="araf.kafka.appender.KafkaAppender">
        <encoder>
            <pattern>${CONSOLE_LOG_PATTERN:-%clr(%d{yyyy-MM-dd
                HH:mm:ss.SSS}){faint} %clr
($ {LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:-
                }){magenta} %clr(---){faint} %clr([%15.15
t]){faint}
                %clr(%-40.40logger{39}){cyan} %clr(:)
{faint}
                %m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}}
</pattern>
            </encoder>
        </appender>

        <logger name="audit-info" level="DEBUG"> category
            <appender-ref ref="kafkaAppender" /> appenderconsolefile
        </logger>

        <root>
            <appender-ref ref="kafkaAppender" />
        </root>
    </configuration>

```

代码中使用

```

@RestController
@Slf4j(topic = "audit-info") loggercategoryname
public class MasGdsController extends AbstractRestController<MasGdsVO,
MasGds>
    implements MasGdsApi {

    @Override
    public Page<MasGdsVO> findAll(DevQueryVO vo) {
        log.debug("=====test slf4j category
level=====");
        return this.doFindAll(vo);
    }
}

```


代码中日志使用

lombok方式

类上增加@Slf4j注解

debug级别日志:

```
        if(log.isDebugEnabled()) {  
            log.debug("begin read file. {}", filePath);  
        }
```