目录

数据接入KES

金山云 Elasticsearch 服务提供在用户 VPC 内通过私有网络和外网两种方式访问集群，您可以通过 Elasticsearch REST client 编写代码访问集群并将自己的数据导入到集群中，也可以通过官方提供的组件（如 logstash 和 beats）接入自己的数据。

本文以官方的 logstash 和 beats 为例，介绍数据接入KES 的方式。

准备工作

如果您在VPC内通过私有网络访问集群，需要创建一台和KES集群相同VPC下的云服务器KEC实例。

使用 logstash 接入 KES 集群

1. 安装部署 logstash 与 java8。请注意 logstash 版本，建议与 Elasticsearch 版本保持一致。

```
wget https://artifacts.elastic.co/downloads/logstash/logstash-5.6.16.tar.gz
tar xvf logstash-5.6.16.tar.gz
yum install java-1.8.0-openjdk  java-1.8.0-openjdk-devel -y
```

2. 根据数据源类型自定义配置文件 *.conf。
    - File 数据源

```
input {
file {
path => "/var/log/nginx/access.log" # 文件路径
}
}
filter {
}
output {
elasticsearch {
hosts => ["http://IP地址:9200"] # KES集群的内网VPC地址和端口
index => "nginx_access-%{+YYYY.MM.dd}" # 自定义索引名称，以日期为后缀，每天生成一个索引
}
}
```

更多有关 File 数据源的接入，请参见官方文档[file input plugin](https://www.elastic.co/guide/en/logstash/5.6/plugins-inputs-file.html)。

- Kafka 数据源

```language
input{
      kafka{
        bootstrap_servers => ["IP地址:6667"] #如果是金山云托管kafka，端口为6667
        client_id => "test"
        group_id => "test"
        auto_offset_reset => "latest" #从最新的偏移量开始消费
        consumer_threads => 5
        decorate_events => true #此属性会将当前 topic、offset、group、partition 等信息也带到 message 中
        topics => ["test1","test2"] #数组类型，可配置多个 topic
        type => "test" #数据源标记字段
      }
}

output {
  elasticsearch {
    hosts => ["http://IP地址:9200"] # KES集群的内网VPC地址和端口
    index => "test_kafka"
  }
}
```

更多有关 kafka 数据源的接入，请参见官方文档kafka input plugin。

使用Beats接入KES集群

Beats 包含多种单一用途的采集器，这些采集器比较轻量，可以部署并运行在服务器中收集日志、监控等数据，相对 logstashBeats 占用系统资源较少。 Beats 包含用于收集文件类型数据的 FileBeat、收集监控指标数据的 MetricBeat、收集网络包数据的 PacketBeat 等，用户也可以基于官方的 libbeat 库根据自己的需求开发自己的 Beat 组件。

KEC中访问KES集群

1. 安装部署filebeat。

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-5.6.16-linux-x86_64.tar.gz
tar xvf filebeat-5.6.16.tar.gz
```

2. 配置filebeat.yml。

```
// 输入源配置
filebeat.prospectors:
- input_type: log
paths:
- /usr/local/services/testlogs/*.log
```

// 输出到 KES output.elasticsearch:

Array of hosts to connect to.

hosts: ["IP地址:9200"]

3. 执行filebeat。

```language
nohup ./filebeat 2>&1 >/dev/null &
```

Filebeat安装部署

Filebeat是本地文件的日志数据采集工具，可监控日志目录或特定日志文件（tail file），并将文件内容转发给Elasticsearch、Logstash、Kafka等。安装包带有部分通用模块，可通过指定命令来简化通用日志格式的收集，解析。

操作步骤

1. 下载对应的版本解压，注意版本要兼容对应的Output组件，具体可参见Download Filebeat。
2. 配置示例文件：filebeat.reference.yml（包含所有配置项）
3. 配置文件：filebeat.yml
4. 测试配置文件是否正确：./filebeat test config
5. 启动：filebeat：./filebeat -e

示例

输出到logstash的示例

```
filebeat.inputs:
- type: log
  enabled: true
  paths:  #配置多个日志路径
    - /var/logs/a.log
    - /var/logs/b.log

output.logstash:
  hosts: [ip1:port1,ip2:port2]
  loadbalance: true  #使用了负载均衡
```

输出到Elasticsearch的示例

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/logs/a.log
    - /var/logs/b.log

filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
output.elasticsearch:
  hosts: ["es1:9200","es2:9200"]
```

Filebeat模块

Filebeat提供了一组预先构建的模块，可以使用这些模块快速实现并部署一个日志监控解决方案，包括数据可视化等，节省了配置的时间。Filebeat modules简化了公共日志格式的收集、解析和可视化。

模块配置文件默认路径：

`${path.config}/modules.d`以及`${path.config}/module`

查看生效模块列表：

```
./filebeat modules list
```

使模块（Elasticsearch）生效：

```
./filebeat modules enable elasticsearch
```

详情请参考：Filebeat Reference [7.10] > Modules。

Logstash部署

根据Elasticsearch的版本下载对应版本的Logstash，假设我们使用6.8.4版本的Elasticsearch。

操作步骤

1.首先从官网下载安装包。（下载地址：https://artifacts.elastic.co/downloads/logstash/logstash-6.8.4.tar.gz）

2.解压安装包。切换目录，执行测试命令：

```
bin/logstash -e 'input { stdin { } } output { stdout {} }'
```

3.等待启动成功，输入任意命令，如`Hello,World`，查看是否正常输出。

入门示例

Logstash主要包含3个部分： 输入(inputs)，过滤器(filters)，输出(outputs)。在有些情况下，我们可以没有过滤器。在过滤器的部分，它可以对数据源的数据进行分析，加工，处理等等。

示例1

1.创建配置文件`heartbeat.conf`。

```
input {
  heartbeat {
    interval => 10
    type => "heartbeat"
  }
}

output {
  stdout {
    codec => rubydebug
  }
```

2.通过`bin/logstash -f heartbeat.conf`启动。

示例2

```
input {
  file {
    path => "/data/*.log"
    start_position => "beginning"
  }
}

filter {
     grok {
        match => { "message" =>      "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
      }
}

output {
 elasticsearch {
     hosts => ["127.0.0.1:9200"]
     index => "test"
 }
}
```

插件详情请参考

input插件：

https://www.elastic.co/guide/en/logstash/6.8/input-plugins.html

output插件：

https://www.elastic.co/guide/en/logstash/6.8/output-plugins.html

filter插件：

https://www.elastic.co/guide/en/logstash/6.8/filter-plugins.html

grok插件：

https://www.elastic.co/guide/en/logstash/6.8/plugins-filters-grok.html

性能调优

操作步骤

1. 确认input或者output吞吐没有到达瓶颈。
2. 查看系统指标比如内存，cpu，io是否有瓶颈。
3. 查看JVM heap是否有压力。
4. 调整pipeline workers 或者 output batch size。

具体细节请参考：https://www.elastic.co/guide/en/logstash/current/performance-troubleshooting.html

数据迁移

Logstash 支持从一个 ES 集群中读取数据然后写入到另一个 ES 集群，因此可以使用 logstash 进行数据迁移，具体的配置文件如下：

```
input {
    elasticsearch {
        hosts => ["http://IP地址:9200"]#数据源
        index => "*"
        docinfo => true
    }
}

output {
    elasticsearch {
        hosts => ["http://IP地址:9200"]#目标端
        index => "%{[@metadata][_index]}"
    }
}
```

上述配置文件将源 ES 集群的所有索引同步到目标集群中，同时也可以设置只同步指定的索引，logstash 的更多功能可查阅logstash 官方文档。

Java API

Elasticsearch5.6.x

Transport Client

1. 创建transport client。

```
// on startup
Settings settings = Settings.builder()
    .put("cluster.name", "myClusterName").build();
//Add transport addresses and do something with the client...
TransportClient client = new PreBuiltTransportClient(Settings.EMPTY)
    .addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName("host1"), 9300))
    .addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName("host2"), 9300));
// on shutdown
client.close();
```

2. 添加参数，client.transport.sniff=true（默认是false，表示不开启集群嗅探功能）。

```
Settings settings = Settings.builder()
        .put("client.transport.sniff", true).build();
TransportClient client = new PreBuiltTransportClient(settings);
```

备注： 这仅是举例说明添加参数的方式，一般client.transport.sniff不需要修改，更多的细节请参考：ES5.6-transport-client。

Elasticsearch6.8.x

强烈建议使用Rest Client，使用Transport Client客户端连接不同版本的elasticsearch实例时，会存在兼容性问题，官方在elasticsearch8.0中不再支持Transport Client。

Transport Client

1. 创建transport client。

```
// on startup
Settings settings = Settings.builder()
    .put("cluster.name", "myClusterName").build();
TransportClient client = new PreBuiltTransportClient(settings)
    .addTransportAddress(new TransportAddress(InetAddress.getByName("host1"), 9300))
    .addTransportAddress(new TransportAddress(InetAddress.getByName("host2"), 9300));
// on shutdown
client.close();
```

2. 添加参数：client.transport.sniff=true（默认是false，表示不开启集群嗅探功能）。

```
Settings settings = Settings.builder()
    .put("client.transport.sniff", true).build();
TransportClient client = new PreBuiltTransportClient(settings);
```

备注： 这仅是举例说明添加参数的方式，一般client.transport.sniff不需要修改,更多的细节请参考ES6.8-transport-client。

Java High Level REST Client

依赖JDK版本1.8+，pom依赖需要添加。

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>6.8.4</version>
</dependency>
```

举个index，get，delete，update的简单例子：

```
RestHighLevelClient client = new RestHighLevelClient(
        RestClient.builder(
                new HttpHost("{elasticsearch集群地址}", 9200, "http")));
//index
Map<String, Object> jsonMap = new HashMap<>();
jsonMap.put("{filed1}", "{value1}");
jsonMap.put("{field2}", "{value2}");
IndexRequest indexRequest = new IndexRequest("{index}", "{type}", "{docid}")
        .source(jsonMap);
IndexResponse indexResponse = client.index(indexRequest , RequestOptions.DEFAULT);
//get
GetRequest getRequest = new GetRequest(
        "{index}",
        "{type}",
        "{docid}");
GetResponse getResponse = client.get(getRequest, RequestOptions.DEFAULT);
```

```
//delete
DeleteRequest deleteRequest = new DeleteRequest(
        "{index}",
        "{type}",
        "{docid}");
DeleteResponse deleteResponse = client.delete(
        deleteRequest, RequestOptions.DEFAULT);
//update
UpdateRequest updateRequest = new UpdateRequest("{index}", "{type}", "{docid}")
        .doc(jsonMap);
UpdateResponse updateResponse = client.update(
        updateRequest, RequestOptions.DEFAULT);
client.close();
```

更多细节请参考官网,ES6.8-Java High Level REST Client。

推荐索引模板

推荐索引模板保存在文件template.json中，内容如下：

```
{
    "order": 0,
    "template": "*",
    "settings": {
        "index": {
            "max_result_window": "65536",
            "routing": {
                "allocation": {
                    "require": {
                        "box_type": "hot"
                    }
                }
            },
            "search": {
                "slowlog": {
                    "threshold": {
                        "query": {
                            "warn": "10s",
                            "info": "5s"
                        }
                    }
                }
            },
            "refresh_interval": "60s",
            "unassigned": {
                "node_left": {
                    "delayed_timeout": "5m"
                }
            },
            "indexing": {
                "slowlog": {
                    "threshold": {
                        "index": {
                            "warn": "10s",
                            "info": "5s"
                        }
                    }
                }
            },
            "number_of_shards": "5",
            "translog": {
                "flush_threshold_size": "5g",
                "sync_interval": "60s",
                "durability": "async"
            },
            "number_of_replicas": "1"
        }
    },
    "mappings": {
        "_default_": {
            "dynamic_templates": [{
                    "all_tpl": {
                        "mapping": {
                            "ignore_above": 1024,
                            "index": "not_analyzed",
                            "type": "{dynamic_type}",
                            "doc_values": true
                        },
                        "match": "*"
                    }
                },
                {
                    "strings": {
                        "match_mapping_type": "string",
                        "mapping": {
                            "type": "keyword"
                        }
                    }
                }
            ],
            "_all": {
                "enabled": false
            }
        }
    },
    "aliases": {}
}
```

模板生效命令：

```
curl -H "Content-Type: application/json" -XPUT
"es_ip:es_port/_template/.default" -d @template.json
```

绑定公网SLB及开放端口

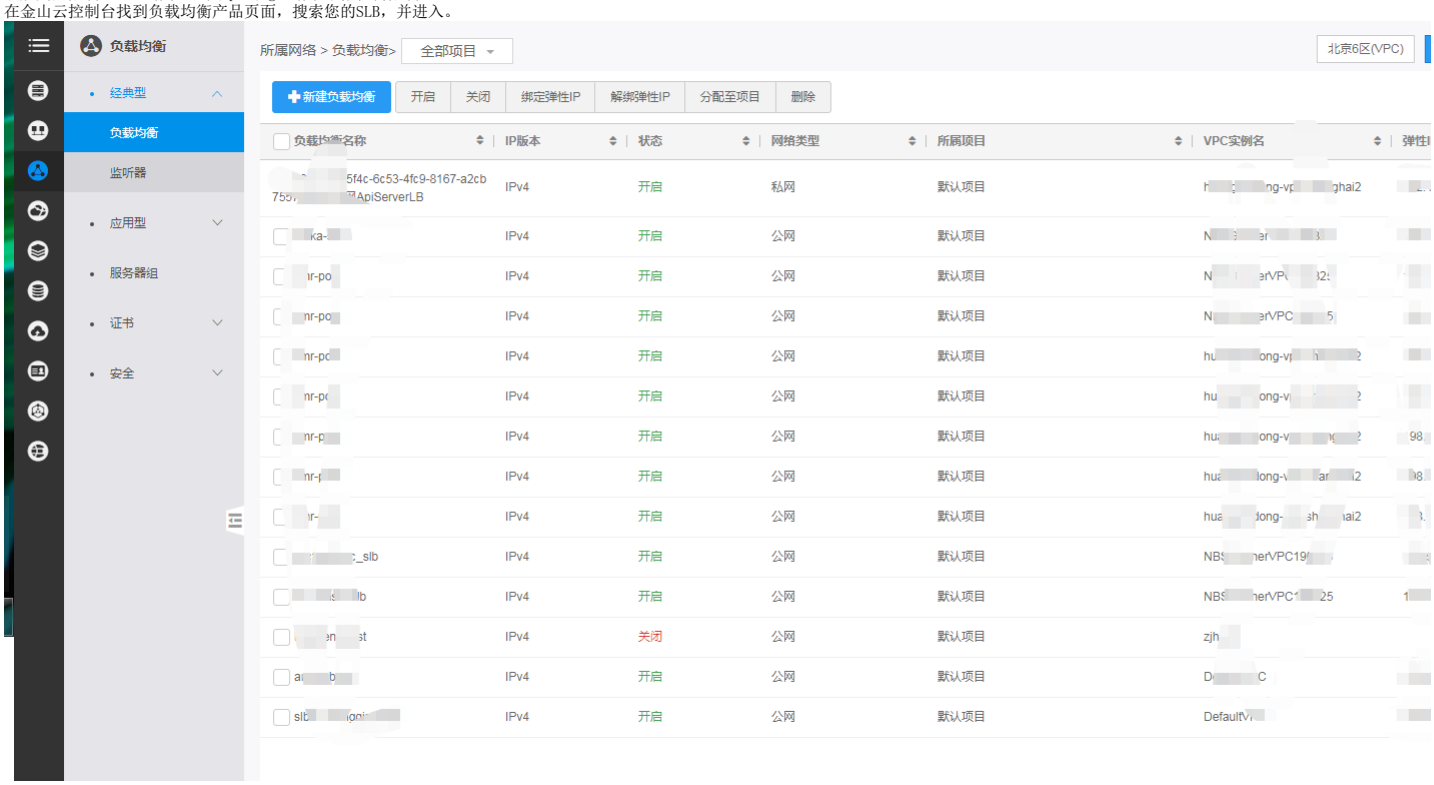
KES绑定公网SLB并且放开端口9200或者9300有集群数据丢失，篡改，泄漏等风险。所以在绑定公网SLB之后，请确保您至少采用以下一种方案来规避风险.

开启X-Pack（推荐）

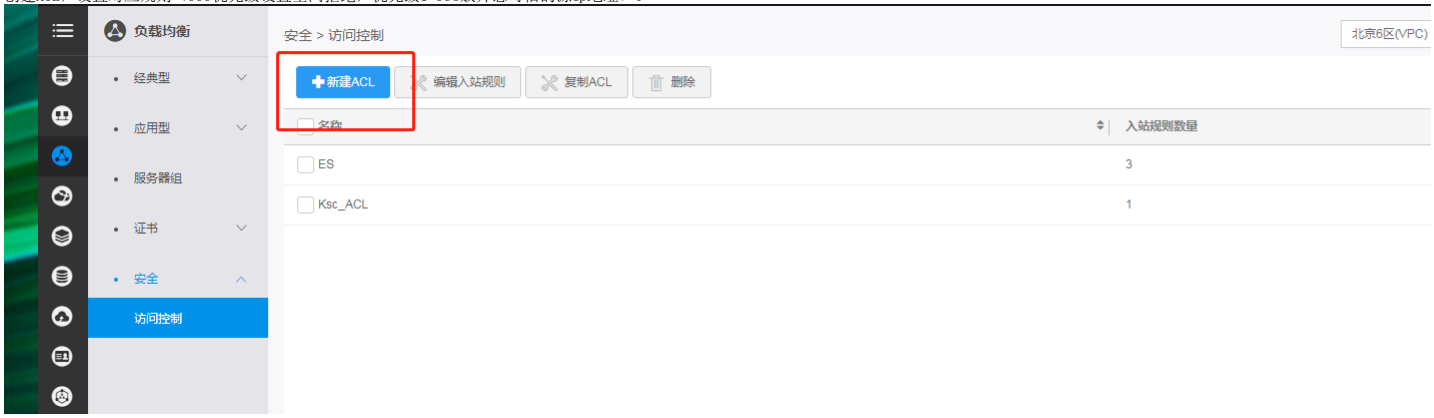
直接通过KES控制台开启，设置访问ES的账户和密码（注意修改历史代码，增加ES访问认证）

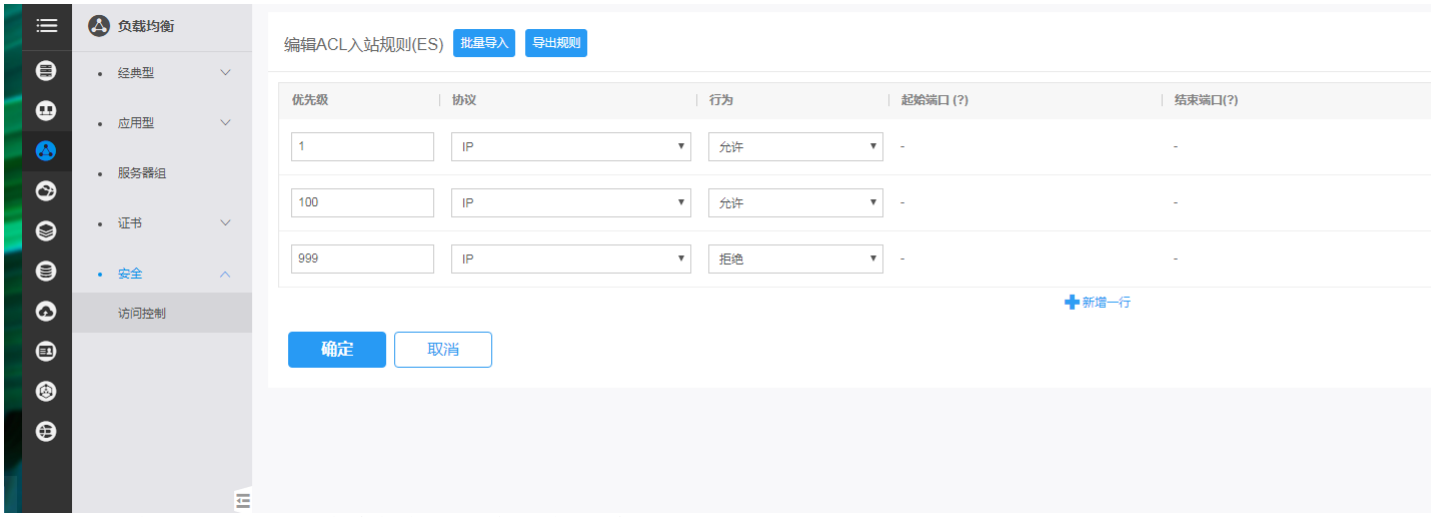
通过SLB的ACL来限制端口访问

该规则只限制delete请求，对于put和get请求不做任何限制。
在金山云控制台找到负载均衡产品页面，搜索您的SLB，并进入。



创建ACL，设置对应规则（999优先级设置全网拒绝，优先级1-998放开您可信的源ip地址）。





创建监听器，监听端口9200，并设置轮训的后端服务端口为19200，开启ACL，绑定上一步设置的ACL。