# 目录

# Java API

### Elasticsearch5.6.x

#### Transport Client

- 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();
```

- 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

- 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();
```

- 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

# 迁移数据

### Logstash

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

# 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

# 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

1）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

2）Kafka 数据源

```
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

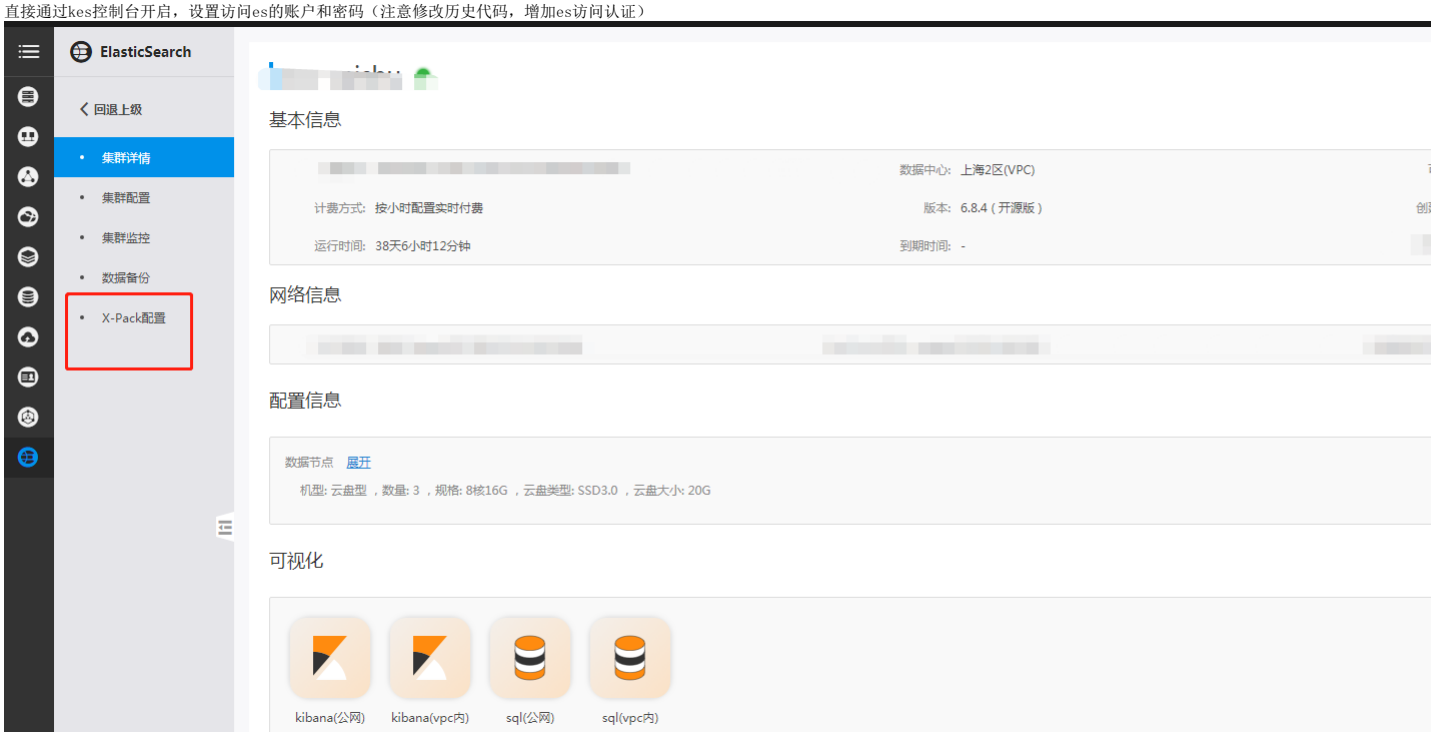
```
nohup ./filebeat 2>&1 >/dev/null &
```

# 绑定公网LB及开放端口

kes绑定公网lb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lb之后，请确保您至少采用以下一种方案来规避风险：

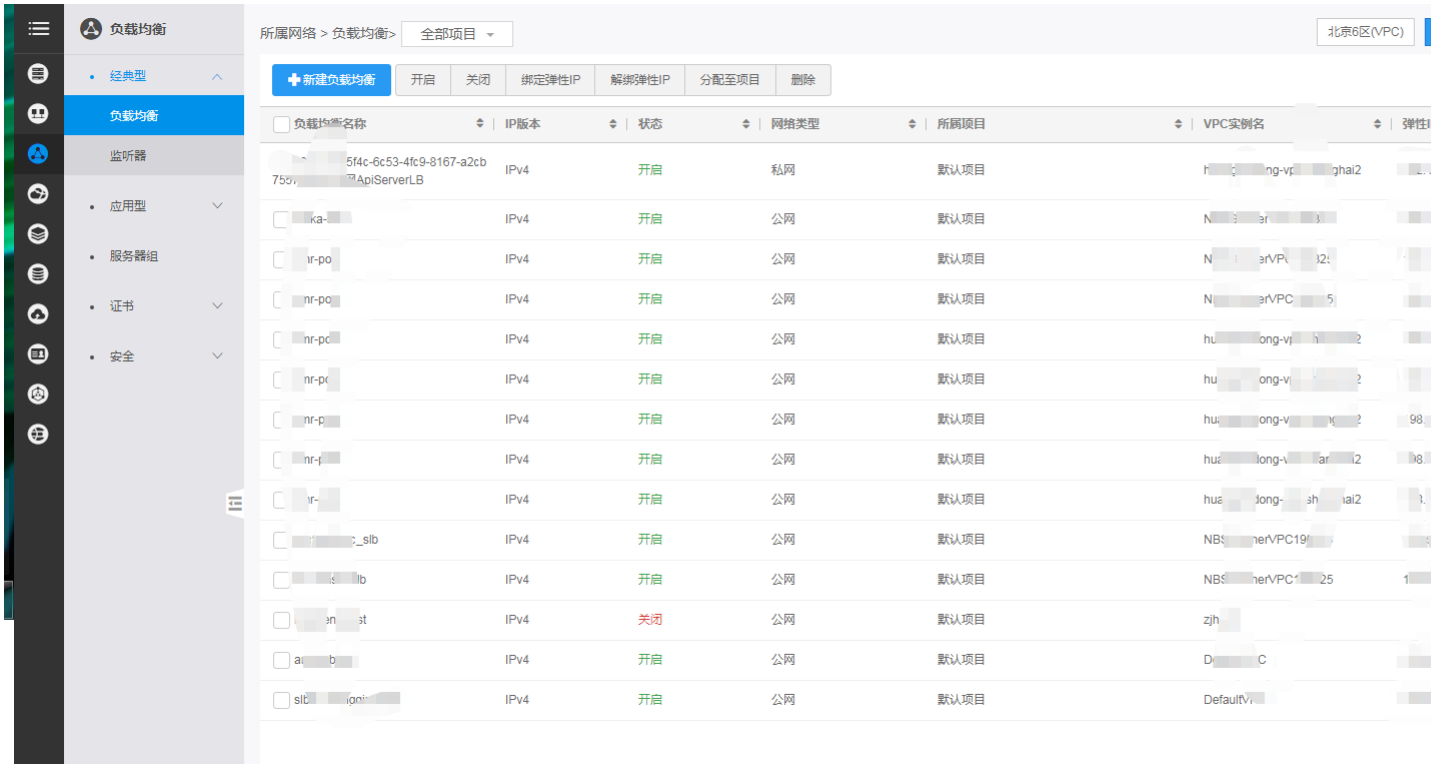
## 开启x-pack（推荐）

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）

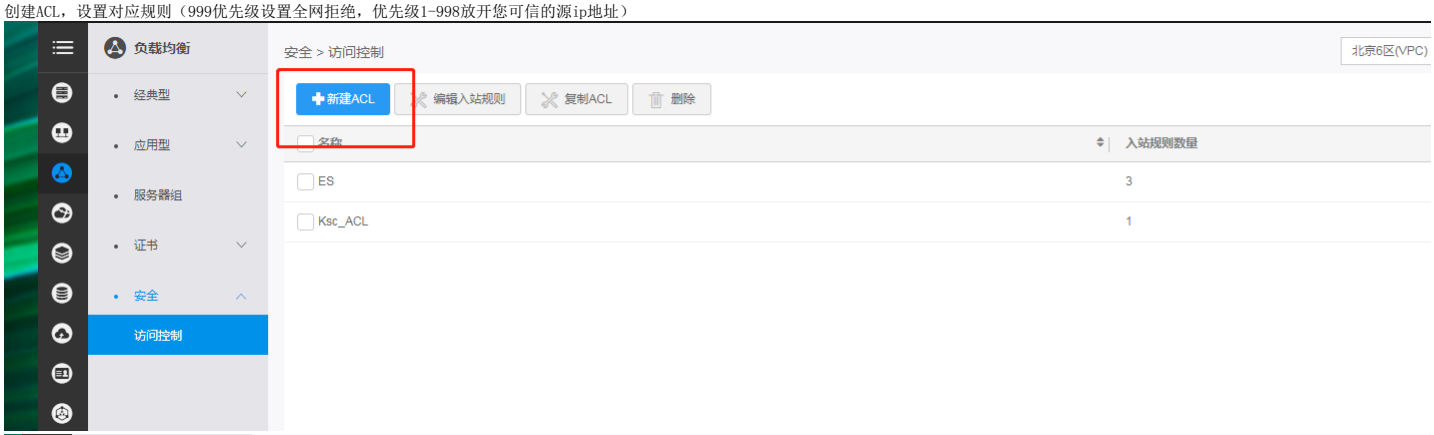


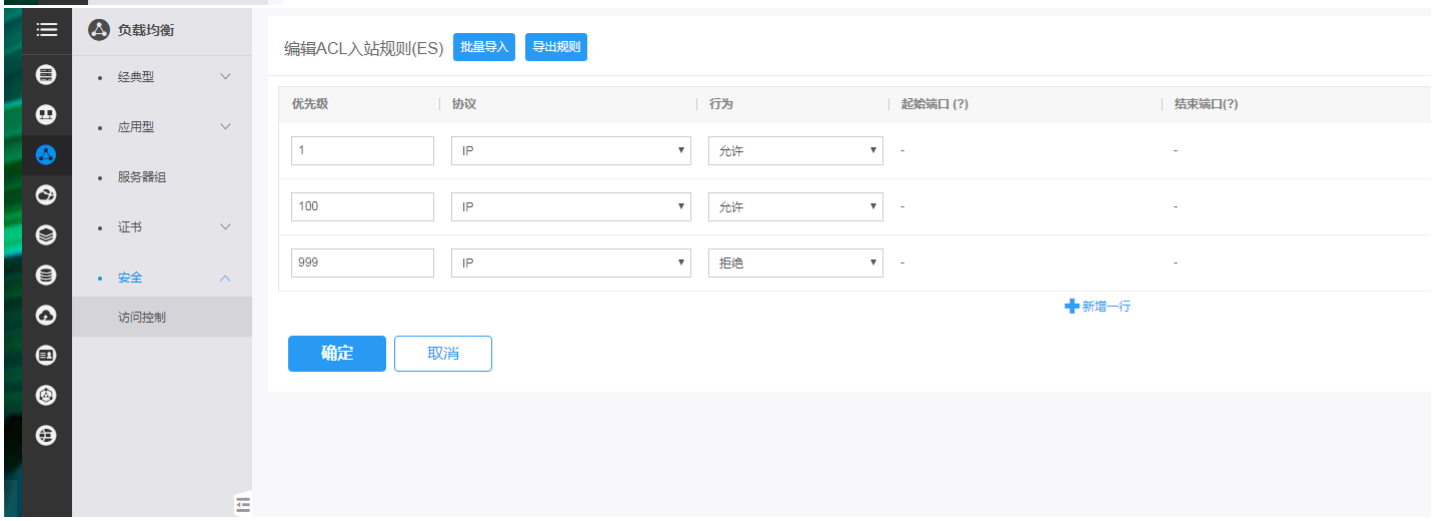

## 通过LB的ACL来限制端口访问

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LB，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