



# 基于Consul的多Beats接入 管控与多ES搜索编排

梁成

腾讯云, [barryliang@tencent.com](mailto:barryliang@tencent.com)



## 拥抱开源、释放云原生的力量

- 背景与挑战
- 多Beats/Logstash接入管控
- 多ES搜索编排系统
- 日志AIOps探索

# 背景与挑战

如何降低日志接入门槛

如何保证日志实时上报

如何保障日志采集不影响业务

如何做配置标准化

如何帮助业务快速排障

如何提供方便便捷的性能分析  
调优能力

...

**100+** 产品数量

**1000** 人员规模

+

**10000** 主机规模

+

# 多Beats/Logstash接入 管控

提供多产品接入管理，多beats标准化、界面化、自动化的日志接入方案

# 案例:1000+业务10000+台 主机如何快速实现日志接入?

## 业务规模

1000+业务、10000+业务主机、每天百T日志增量

## 日志需求

收集业务日志文件用于故障分析与告警监控

收集主机性能数据做容量分析

日志热数据保存七天

历史数据冷备一个月

## 其他诉求

日志上报不能影响核心业务

数据上报延时可感知

## 传统Beats接入流程

现网配置是否全部一致?

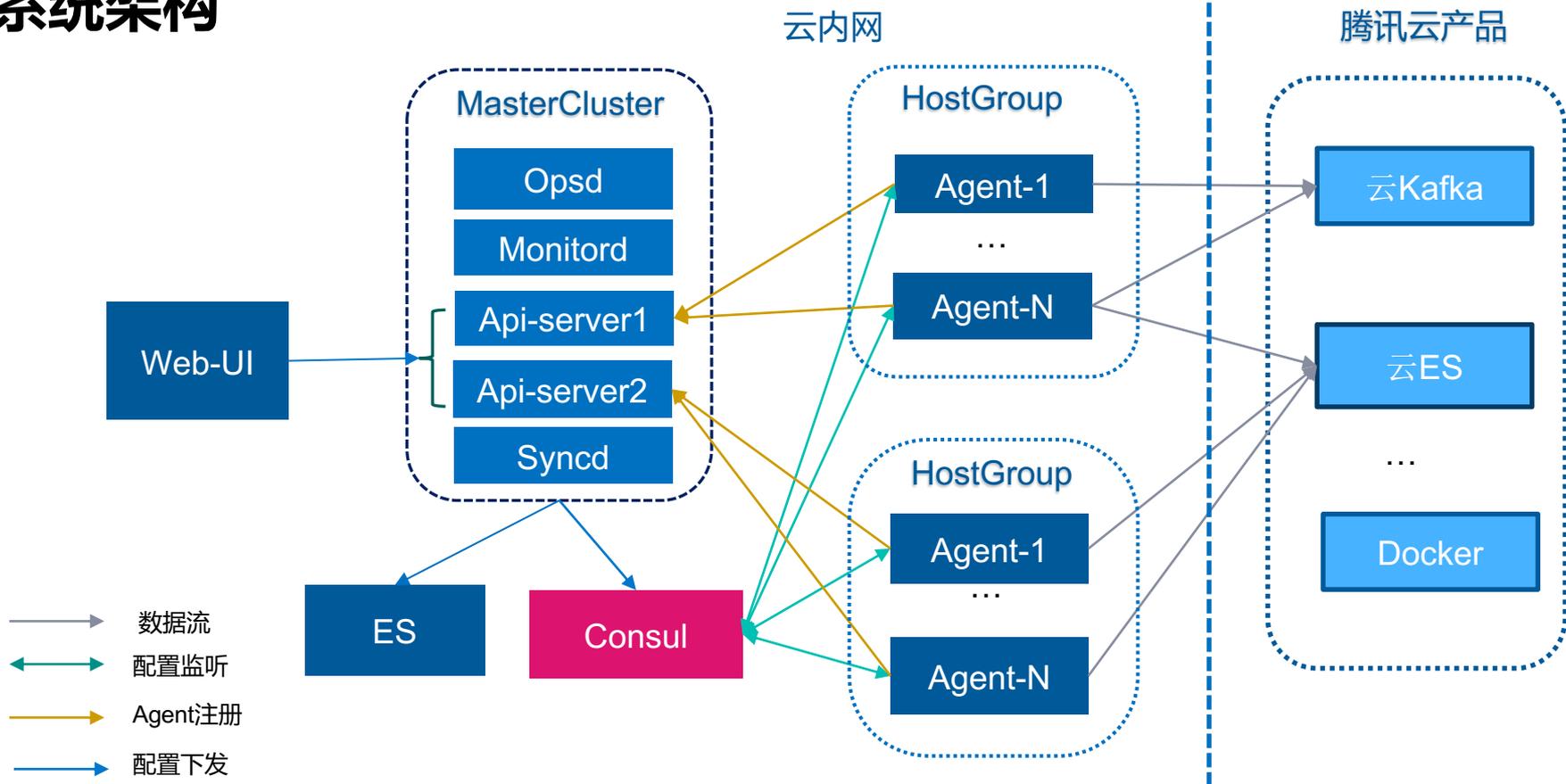
日志上报是否有延时?

Filebeat是否资源消耗过多?

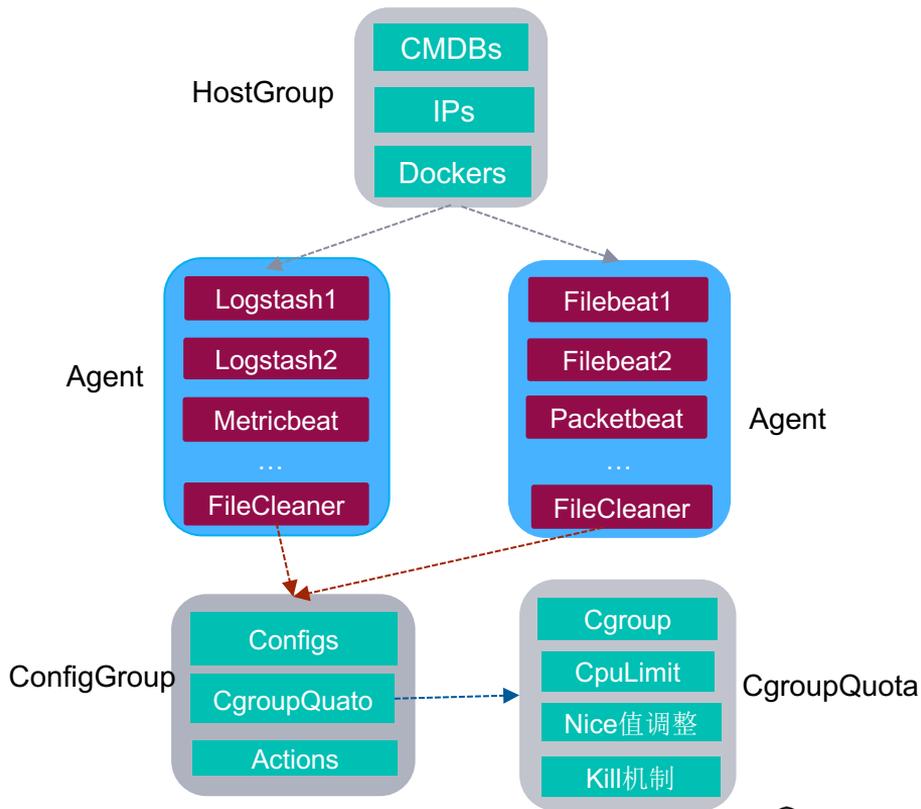
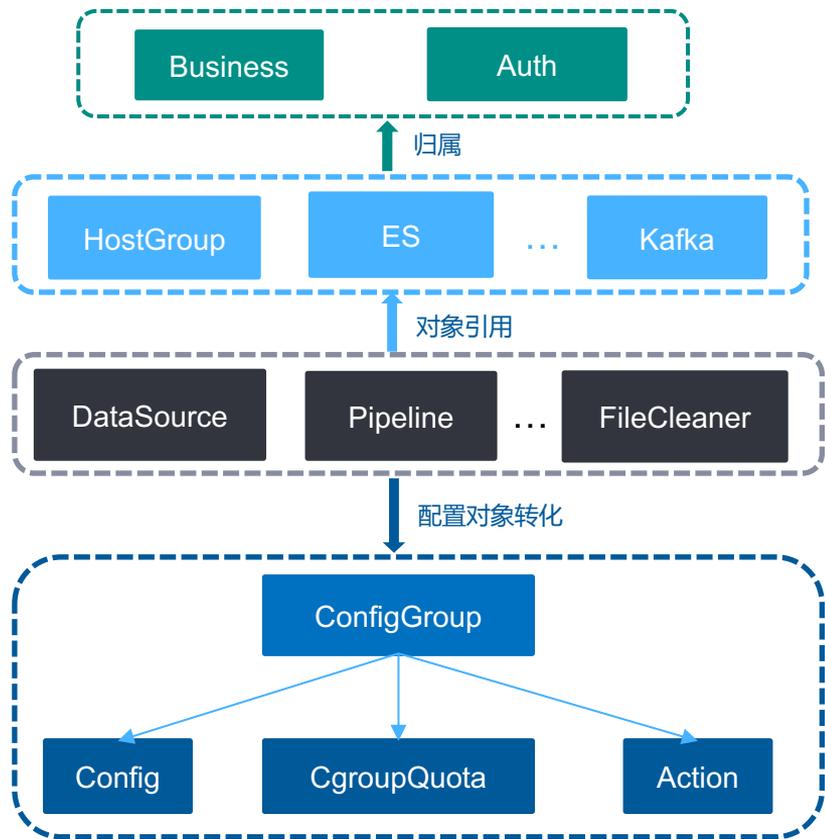
Filebeat异常退出如何  
处理?

如何做上报性能调优?

# 系统架构



# 数据模型



# Agent管理

## 时序图

### Agent注册

Agent启动首先向Consul获取Master服务列表，并向Master发起Agent注册逻辑，获取agent id

### 配置获取

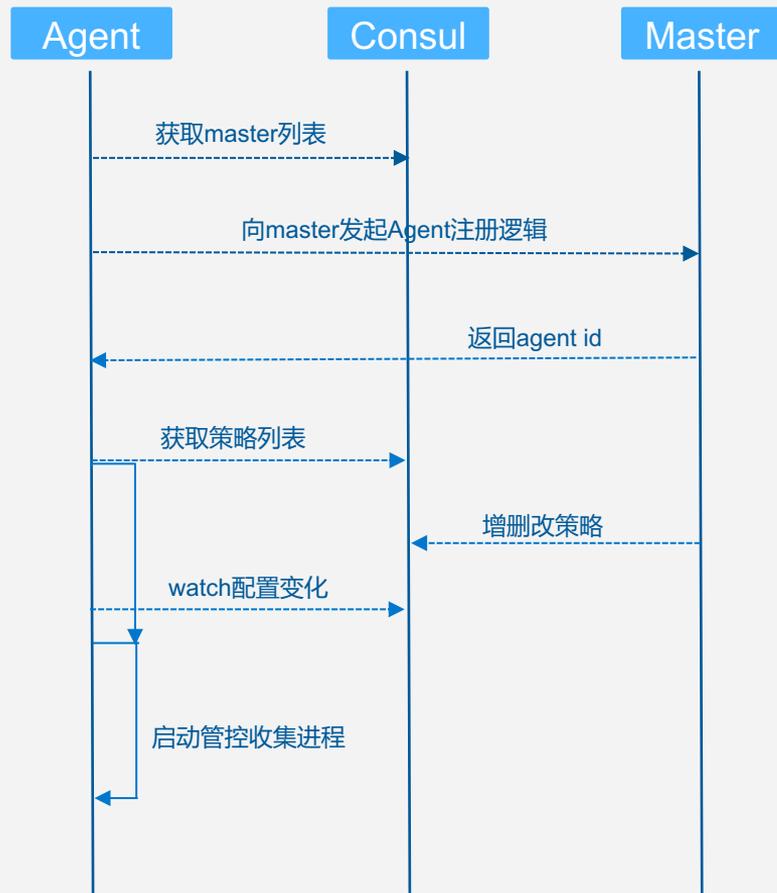
从Consul中获取当前agent的配置组列表，并启动多个采集进程

### 配置变更感知

watch到Consul对应的agent id路径，实时感知配置变化，并对启动的进程列表做重启清理等工作

### 管理多Beats/logstash

Beats等以agent子进程启动其管理这些进程的cpu/内存等资源



```
[root@~]# cat /data/log_ops_agent/qcloud_fbeat
/3909bd45b00a5b1/config/qcloud_fbeat.yml
fields:
  name:
  region: ${LOG_OPS_REGION:none}
  report_ip: ${LOG_OPS_IP:none}
fields_under_root: true
filebeat.inputs:
- type: log
  enable: true
  paths:
  - /*.log
  tail_files: true
  ignore_older: 24h
  clean_inactive: 25h
  harvester_limit: 100
  fields_under_root: true
  scan_frequency: 2s
  harvester_buffer_size: 1638400
  max_bytes: 2485760
- type: log
  enable: true
  paths:
  - /*.log
  tail_files: true
  ignore_older: 24h
  clean_inactive: 25h
  harvester_limit: 100
  fields_under_root: true
  scan_frequency: 2s
  harvester_buffer_size: 1638400
  max_bytes: 2485760
- type: log
  enable: true
  paths:
  - /*.log
  tail_files: true
  ignore_older: 24h
  clean_inactive: 25h
  harvester_limit: 100
  fields_under_root: true
  scan_frequency: 2s
  harvester_buffer_size: 1638400
  max_bytes: 2485760
```

```
1 {
2   type: "log",
3   "id": "3909bd45b00a5b1",
4   "version": 3,
5   name: "price_业务日志",
6   "description": "jf_price_log",
7   "version_info": "",
8   create_at: "2019-02-21T11:02:21+08:00",
9   "modified_at": "2019-09-03T10:54:51.604607675+08:00",
10  "auth": {
11    "creator": "admin",
12    "owners": [
13      "admin"
14    ],
15    "readers": null
16  },
17  "configs": [
18    {
19      "json_values": "{}",
20      "template": "fields:\n name jf_price_log\n region: ${LOG_OPS_REGION:none}\n report_ip: ",
21      "meta": {
22        "file": "/data/log_ops_agent/qcloud_fbeat/3909bd45b00a5b1/config/qcloud_fbeat.yml",
23      },
24      "content": "fields:\n name: jf_price_log\n region: ${LOG_OPS_REGION:none}\n report_ip: ",
25    },
26    {
27    }
28  ],
29  "pre_actions": [
30    {
31      "exe": "mkdir",
32      "args": [
33        "-p",
34        "/data/log_ops_agent/qcloud_fbeat/3909bd45b00a5b1/monitor"
35      ]
36    },
37  ],
38  "actions": [
39    {
40      "exe": "qcloud_fbeat",
41      "args": [
42        "-c",
43        "/data/log_ops_agent/qcloud_fbeat/3909bd45b00a5b1/config/qcloud_fbeat.yml",
44      ]
45    },
46  ],
47  "cgroup_quota": {
48    "cpu_percent": 10,
49    "mem_quota_type": "percent"
50  }
51 }
```

配置组 id 与版本

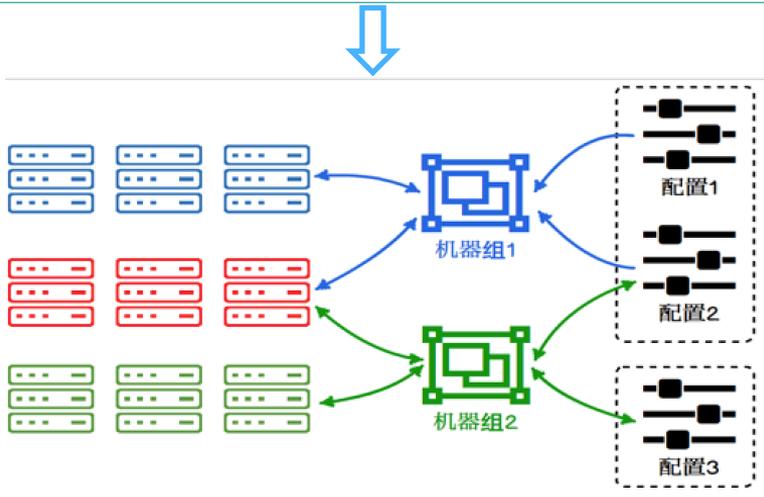
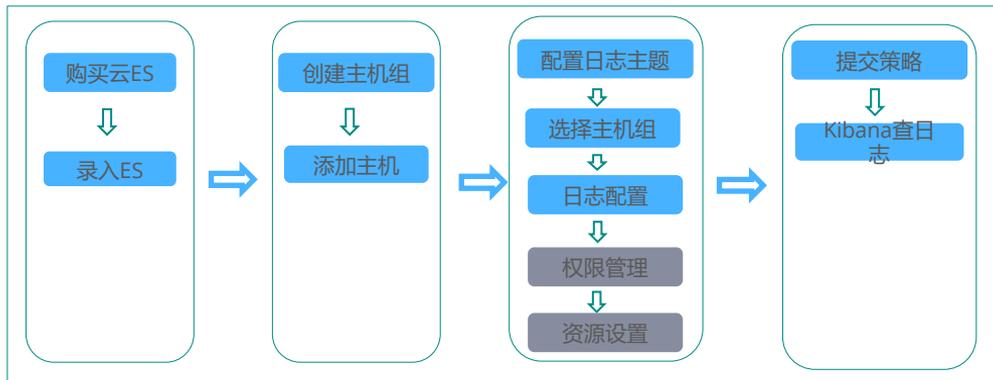
配置文件列表

生成配置前的预处理命令

程序启动命令列表

当前配置组对应的资源配额

# 日志接入



Cancel

提交



# 当前收益

快

- 快速接入(5min)
- 配置UI化标准化
- 配置变更实时感知
- 部署全自动化
- 多Beats支持

稳

- Beats运行时cpu/mem可控
- Agent监控视图
- 离线/容量/延时监控
- 分布式集群管理
- 异常快速定位

准

- 关联公司CMDB
- 资源权限管理
- 配置灰度控制发布
- 配置一致性检测
- 日志覆盖率

# 案例:如何管控整个日志数据流相关资源性能与容量?

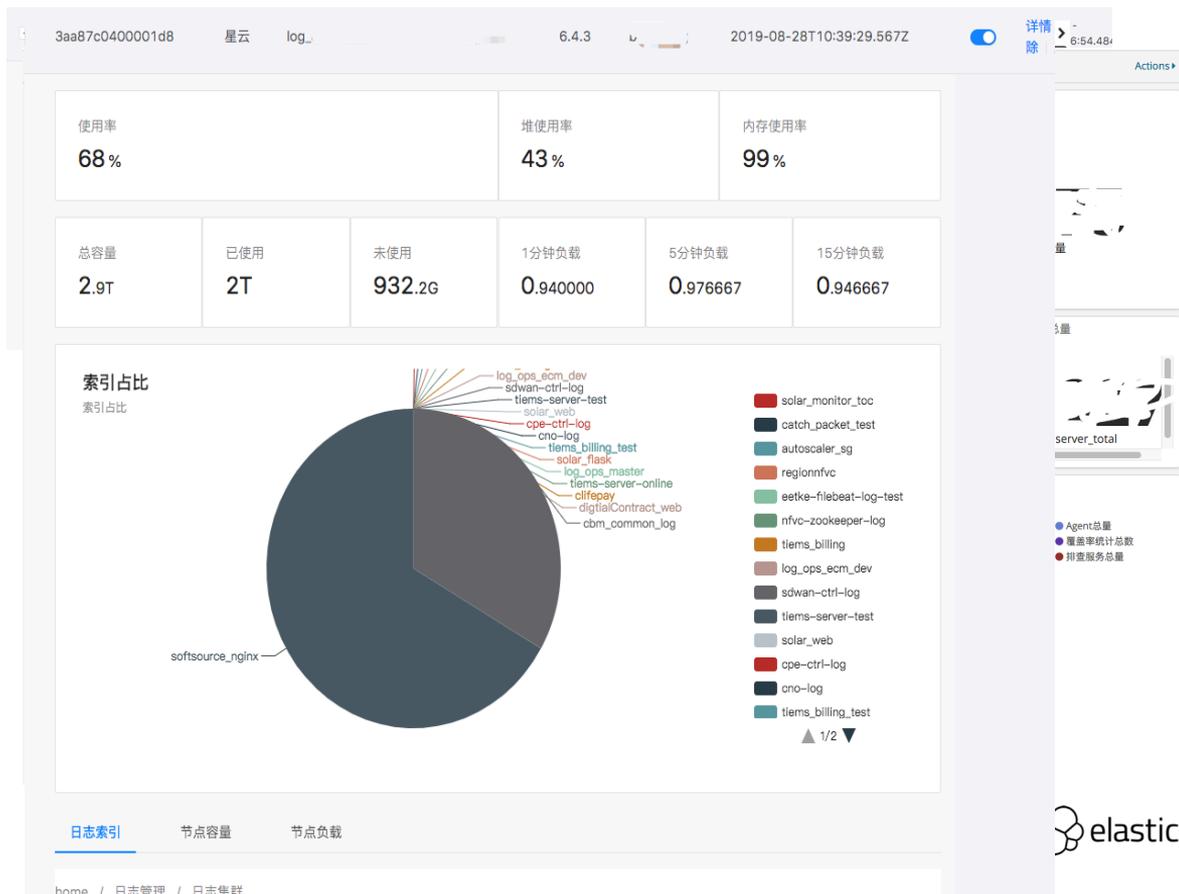
Agent运行时监控

Beats cpu/mem管控

日志延时分析

日志覆盖率分析

ES/kafka容量管理



# 案例:高并发写入场景下Beats与ES性能优化

对比	自建	日志平台接入
上报延时分析	无法实时感知日志积压情况	日志延时dashboard快速定位
Filebeat性能分析	依赖白金版monitor机制	不依赖es版本、结合cpu/mem限额配置与实时指标采集分析
Filebeat性能调优	需要修改配置文件不断尝试	界面提交核心参数并结合延时图对比分析
Filebeat性能管控	日志量太大Cpu飙升影响业务	精准控制资源消耗防止异常减少抖动
Es写入性能调优	修改配置文件不断观察数据情况	基于ES压测报告给出专家级es参数优化建议
参数优化体验	修改配置文件、参数调优相对麻烦	全UI化、一站式处理

# 配置UI化

## 配置UI化开发思路

### 嵌套式表单

大表单套小表单，所有表单都是以angular组件形式开发，保证代码的可复用性与质量

### 配置分级展现

把复杂配置独立成高级选择，并设置默认值，并在复杂项给出有效帮助

### 自定义组件

编写大量自定义小组件，比如cmdb设置，时间设置等组件提高用户体验，尽量减少直接填写文本

### 前后端强类型

前端采用基于ts的angular开发所有数据定义与后端golang数据保持强一致，保证数据一致性

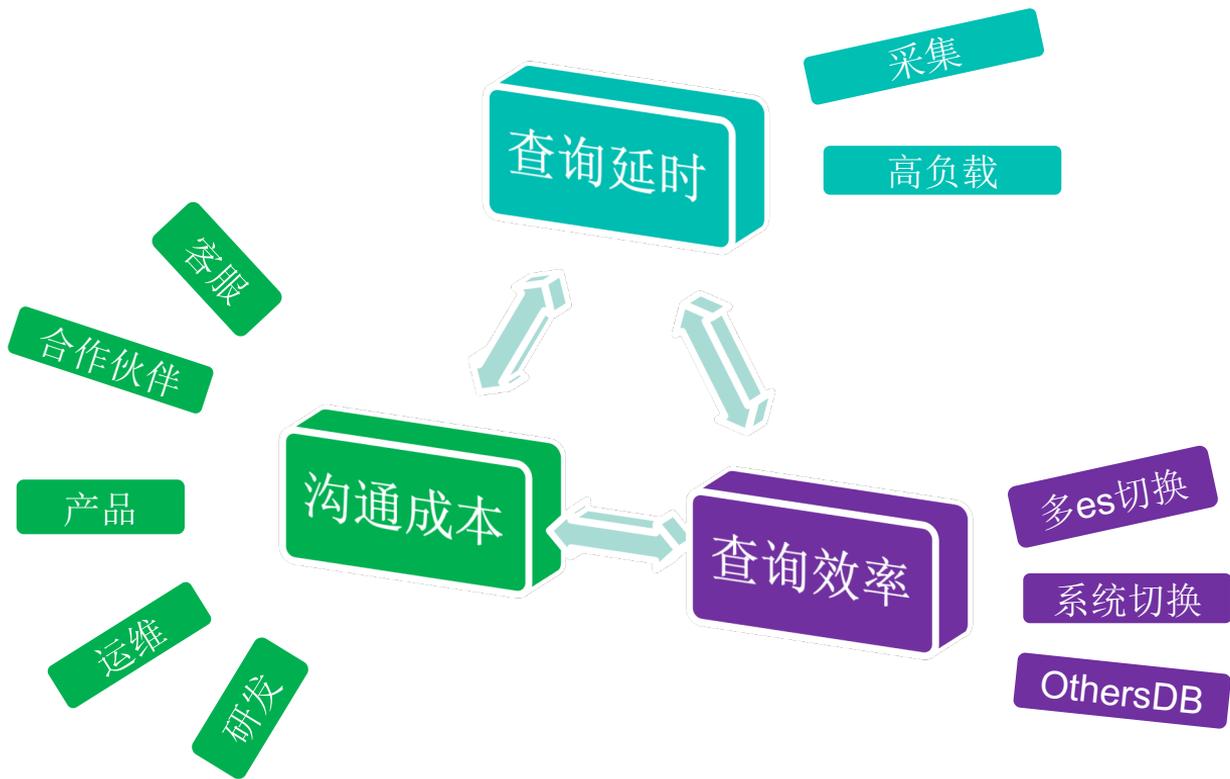
数据源前端ts类定

```
// ElasticSearchWriteConfig ES写入配置
type ElasticSearchWriteConfig struct {
    source-detail.component.html — log_ops
    controller TS source-detail.component.ts source-detail.component.html x 数据源定义模板文件 TS elasticsearch-detail.component.html
    routes > logops > pages > data-source > source-detail > source-detail.component.html
    <!-- the same view can be shown in more than one case -->
    <div *ngSwitchCase="!es!">
        <app-elasticsearch-index-detail [(dataItem)]="dataItem.es_index" [editable]="false" [isSimple]="true"> </app-elasticsearch-index-detail>
        <nz-collapse-panel [nzBordered]="false">
            <nz-collapse-panel nzHeader="高级配置" [nzActive]="esWriteConfigClose"> 索引设置表单组件
                <app-write-es-config [(config)]="dataItem.write_config.es_write"> </app-write-es-config>
            </nz-collapse-panel>
        </nz-collapse-panel>
    </div>
    <div *ngSwitchCase="!kafka!">
        <app-kafka-topic-detail [(dataItem)]="dataItem.kafka_topic" [editable]="false" [isSimple]="true"> </app-kafka-topic-detail>
        <nz-collapse-panel [nzBordered]="false">
            <nz-collapse-panel nzHeader="高级配置" [nzActive]="esWriteConfigClose"> kafka配置表单组件
                <app-write-kafka-config [(config)]="dataItem.write_config.kafka_write"> </app-write-kafka-config>
            </nz-collapse-panel>
        </nz-collapse-panel>
    </div>
    </div>
    </nz-collapse-panel>
</collapse>
collapse [nzBordered]="false">
    <nz-collapse-panel nzHeader="采集对象" nzActive="true" [nzStyle]="hostGroupDetailStyle">
        <app-host-group-detail [(detailVisible)]="detailVisible" [dataItem]="dataItem.host_group" [isSimple]="true"></app-host-group-detail>
    </nz-collapse-panel>
</collapse>
collapse [nzBordered]="false">
    <nz-collapse-panel nzHeader="接入配置" nzActive="true">
        <nz-form-item>
            <nz-form-label [nzSpan]="7" nzRequired="接入方式"></nz-form-label>
            <nz-form-control [nzSpan]="12">
                <nz-select>
                    formControlName="access_way"
                    nzPlaceholder="选择数据源对应业务"
                    [(ngModel)]="dataItem.access_config.access_way"
                >
                <nz-option nzValue="log_file" nzLabel="日志文件"></nz-option>
                <nz-option nzValue="packet" nzLabel="网络抓包"></nz-option>
                <nz-option nzValue="metric" nzLabel="指标获取"></nz-option>
            </nz-select>
        </nz-form-control>
    </nz-form-item>
    <nz-divider nzText="" nzOrientation="left"></nz-divider>
    <div [ngSwitch]="dataItem.access_config.access_way">
    <!-- the same view can be shown in more than one case -->
    <div *ngSwitchCase="log_file"> 文件采集表单组件
        <nz-form-item ngForm="let log_access_config of dataItem.access_config.log_access_items; let i = index">
            <app-access-log-config [logPathConfigItem]="log_access_config" (removeItem)="onRemoveItem($event)"> </app-access-log-config>
        </nz-form-item>
    </div>
    gatherGroupId string json:"g
    writeConfig WriteConfig json:"w
    cgroupQuota CgroupQuota json:"c
    monitorEnable bool json:"m
    public gather_group_id: string, // 定义数据接入的采集
    public write_config: WriteConfig, // 数据写入配置
    public cgroup_quota: CgroupQuota,
```

# 多ES搜索编排系统

提供多ES多索引搜索编排功能，帮助  
业务快速定位异常

# 故障定位遇到的困扰



# 案例:非APM场景下多组件日志搜索探索

The screenshot shows an Elastic search interface with a central diagram and numbered callouts (1-8) highlighting key features:

- 1 多组件搜索**: Multi-component search, indicated by a red arrow pointing to the search results table.
- 2 kibana导航**: Kibana navigation, indicated by a red arrow pointing to the 'kibana导航' button in the search results.
- 3 精准匹配**: Precise matching, indicated by a red arrow pointing to the '精准匹配' button in the search results.
- 4 常规上下文分析**: Regular context analysis, indicated by a red arrow pointing to the '上下文' button in the search results.
- 5 智能打标和分析**: Smart tagging and analysis, indicated by a red arrow pointing to the '智能分析' button in the search results.
- 6 专家场景沉淀**: Expert scenario accumulation, indicated by a red arrow pointing to the search results table.
- 7 关键字染色快速发现错误日志**: Keyword coloring to quickly discover error logs, indicated by a red arrow pointing to the search bar.
- 8 ELK原生语法兼容**: ELK native syntax compatibility, indicated by a red arrow pointing to the search bar.

The central diagram features a central circle labeled 'ES/ OtherDB' surrounded by five other circles: '多集群多索引' (Multi-cluster, Multi-index), '上下文搜索' (Context search), 'Kibana 导航' (Kibana navigation), '搜索编排' (Search orchestration), and '异常知识库' (Anomaly knowledge base).

The search interface includes a search bar with the query '4nv77sek', a search button, and a results table with columns for source, match status, and count. The results table shows:

Source	Match Status	Count
腾讯云-qcbase	匹配到	[1] 条数据
腾讯云-云API	匹配到	[32] 条数据
腾讯云-云	匹配到	[0] 条数据
VPC-vpc_api_v3	匹配到	[71] 条数据

# 日志AIOps探索

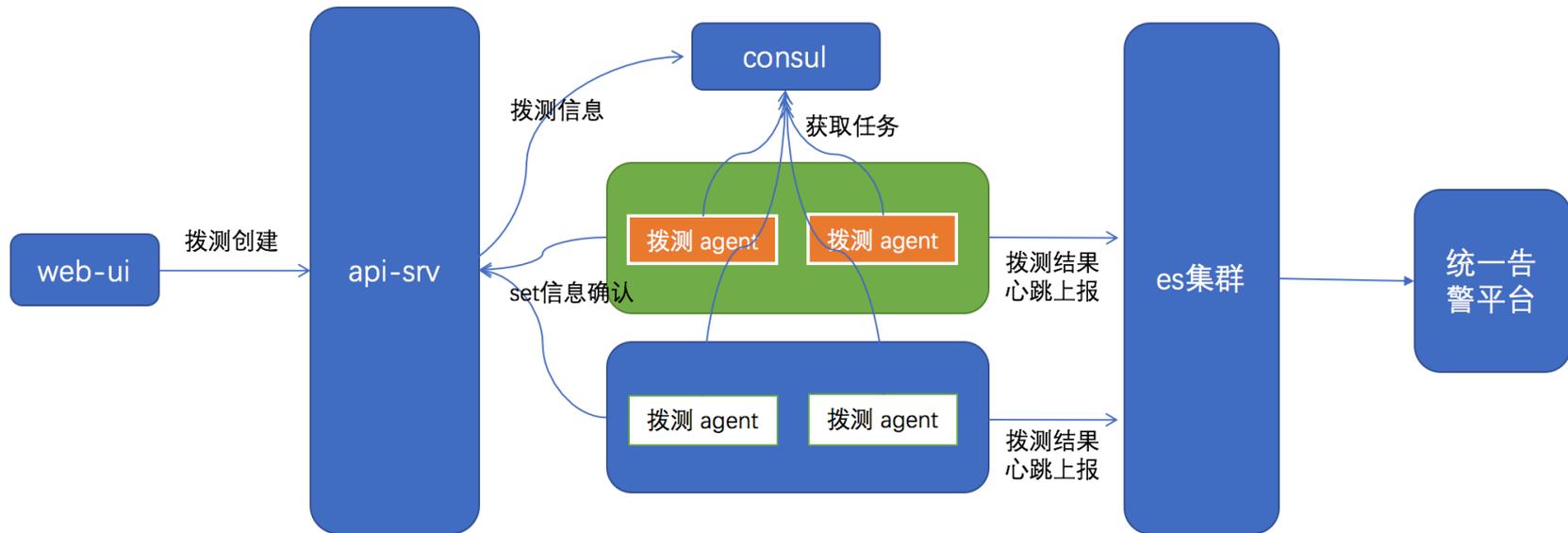
日志最佳实践、拨测系统、智能排障、  
AIOps探索

# 案例: 基于Kibana的交互式排障



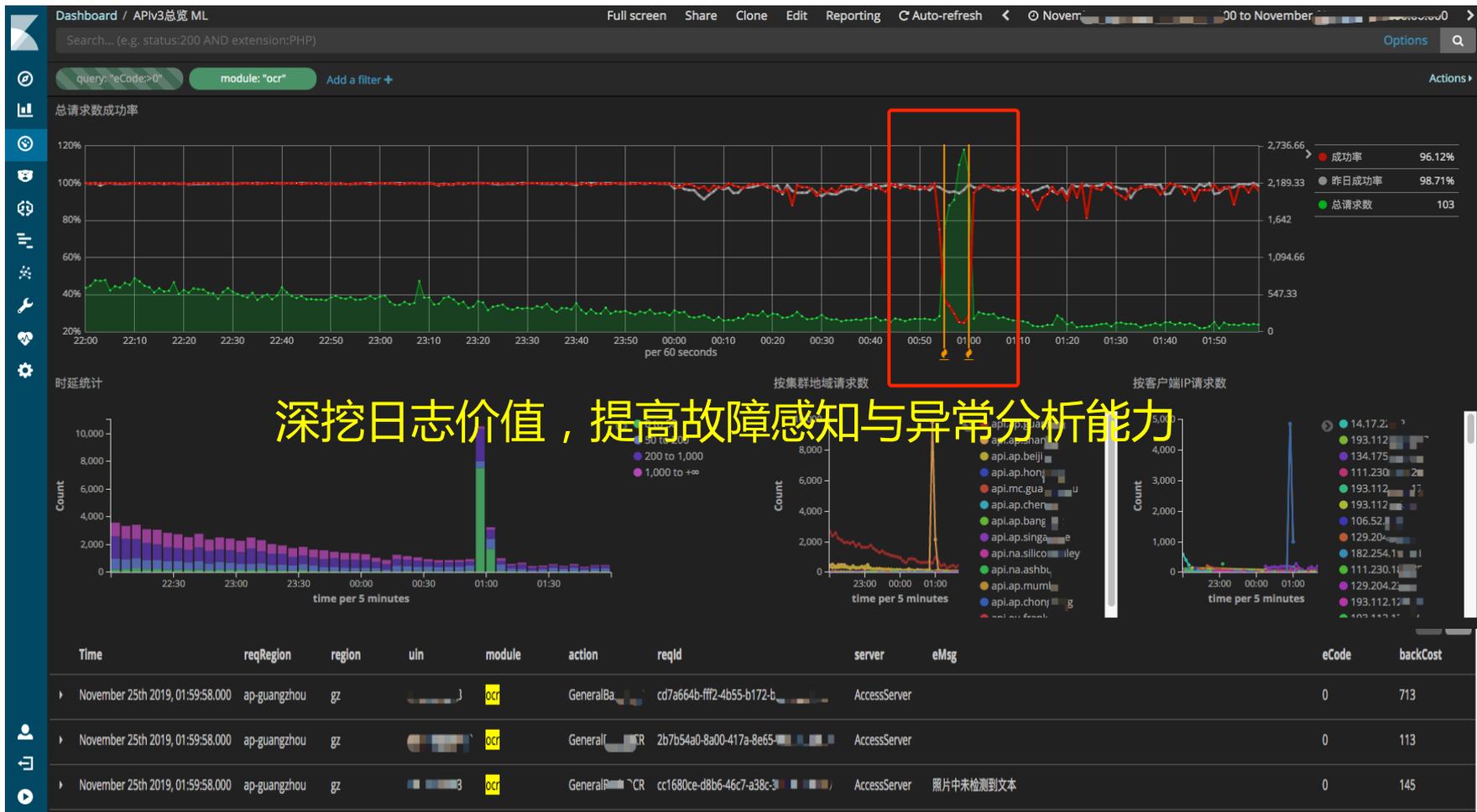
交互式排障, 下钻分析, 对比分析, 快速定位异常

# 案例:如何基于ELK构建内网拨测系统?



基于Healthbeat与Metricbeat构建分布式、全region覆盖、协议支撑丰富的内网拨测平台

# 案例: 使用ES原生ML能力提供排障效率



# 技术栈

## 技术选型

### 前端

TypeScript、Angular 8、Ng-zorro

### 后台

Golang、Iris、Consul、微服务化设计、Pongo2模板管理

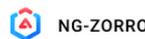
### 配置

Consul、Elasticsearch

### Agent

Golang、ELK beats、Logstash、LXC资源管  
控、  
Consul、其他自定义插件

前端



后台



配置



Agent



谢谢大家！