B站日志系统的演进之路

王翔宇

目录

B站的日志系统基于elastic stack，面向全站提供统一的日志服务。

- 日志规范的制定及日志系统1.0的实现
- 日志系统1.0的精细化运维
- 日志监控功能的实现
- 日志系统1.0+的多集群建设
- 展望

日志系统的设计目标

- 原有问题：
  - 方案各异, 可靠性差
  - 业务日志没有统一的规范
  - 对PAAS支持不好
  - 日志使用成本高
- 设计目标：
  - 收敛接入方式，对PAAS友好
  - 规范日志格式
  - 日志解析对日志系统透明
  - 系统高可用、容量可扩展、高可运维性

日志格式规范

- JSON作为日志的输出格式
- 必须包含四类元信息：
  - time: 日志产生时间，ISO8601格式
  - level：日志等级， FATAL、ERROR、WARN、INFO、DEBUG
  - app_id：应用id，用于标示日志来源，与服务树一致，部门.项目.应用 三级，全局唯一
  - instance_id：实例id，用于区分同一应用不同实例，业务方自行设定
- json的mapping应保持不变：key不能随意增加、变化，value的类型也应保持不变。


```
{
  "time": "2018-06-18T03:00:00.002Z",
  "app_id": "main.community.dm2",
  "instance_id": "dm2-5175-698475c848-qp5gq",
  "level": "INFO",
  "log": " request /x/web-feed/feed/unread succeed",
  "thread_name": "pool-1-thread-245",
  "req": {
    "bucket": "upgcxcode",
    "mission_id": "150e5feb66a6242554d4d8248481b35c",
    "msg": "celery:root_callback:called",
    "request_id": "c9d9992f-527a-4e79-b176-0763d4408c8d",
    "sid": "50001",
  }
}
```

日志系统1.0的实现

- 日志从产生到可检索，经历四个阶段：
  - 采集
  - 传输
  - 切分
  - 存储和检索

日志系统1.0-采集

- 采集（三种）

log agent:

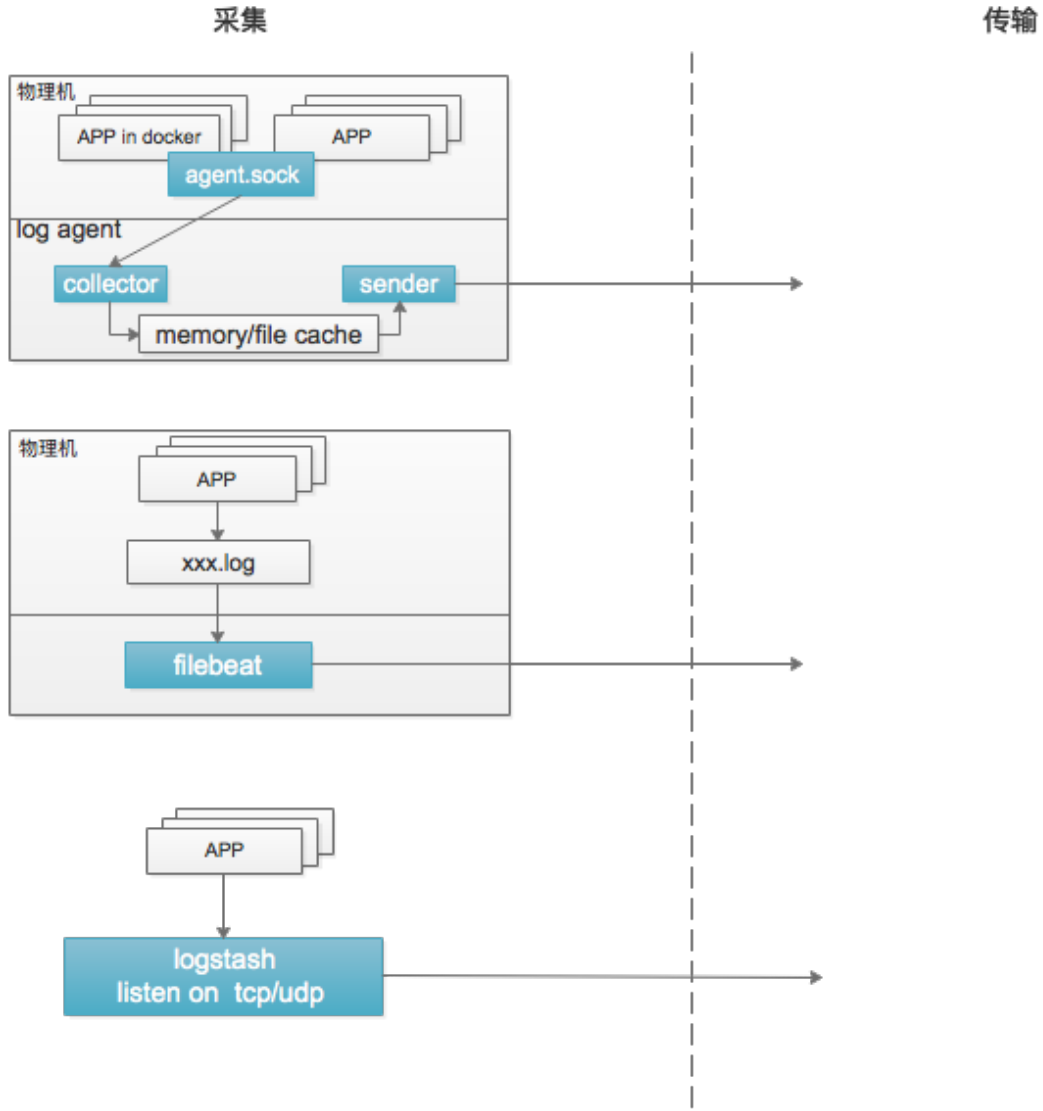
- 物理机部署，监听sock，应用写sock完成日志输出
- 对于paas友好
- 日志系统提供各种语言SDK。
- 适用于的自研类应用：按照日志规范输出日志

logstash:

- 监听tcp/udp，
- 适用于通过网络上报日志的方式

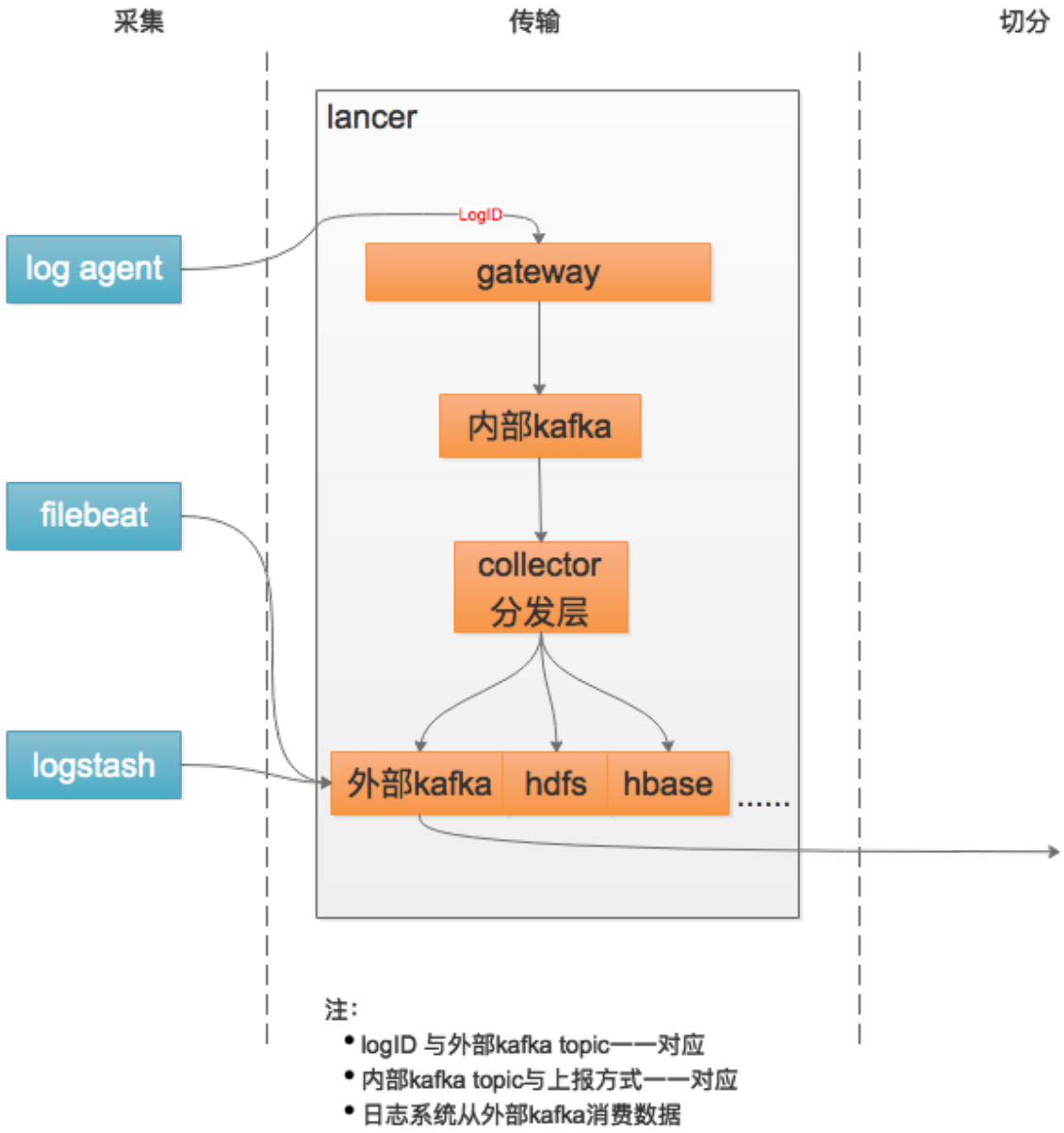
filebeat:

- 直接采集本地生成的日志文件
- 适用于日志无法定制化输出的应用



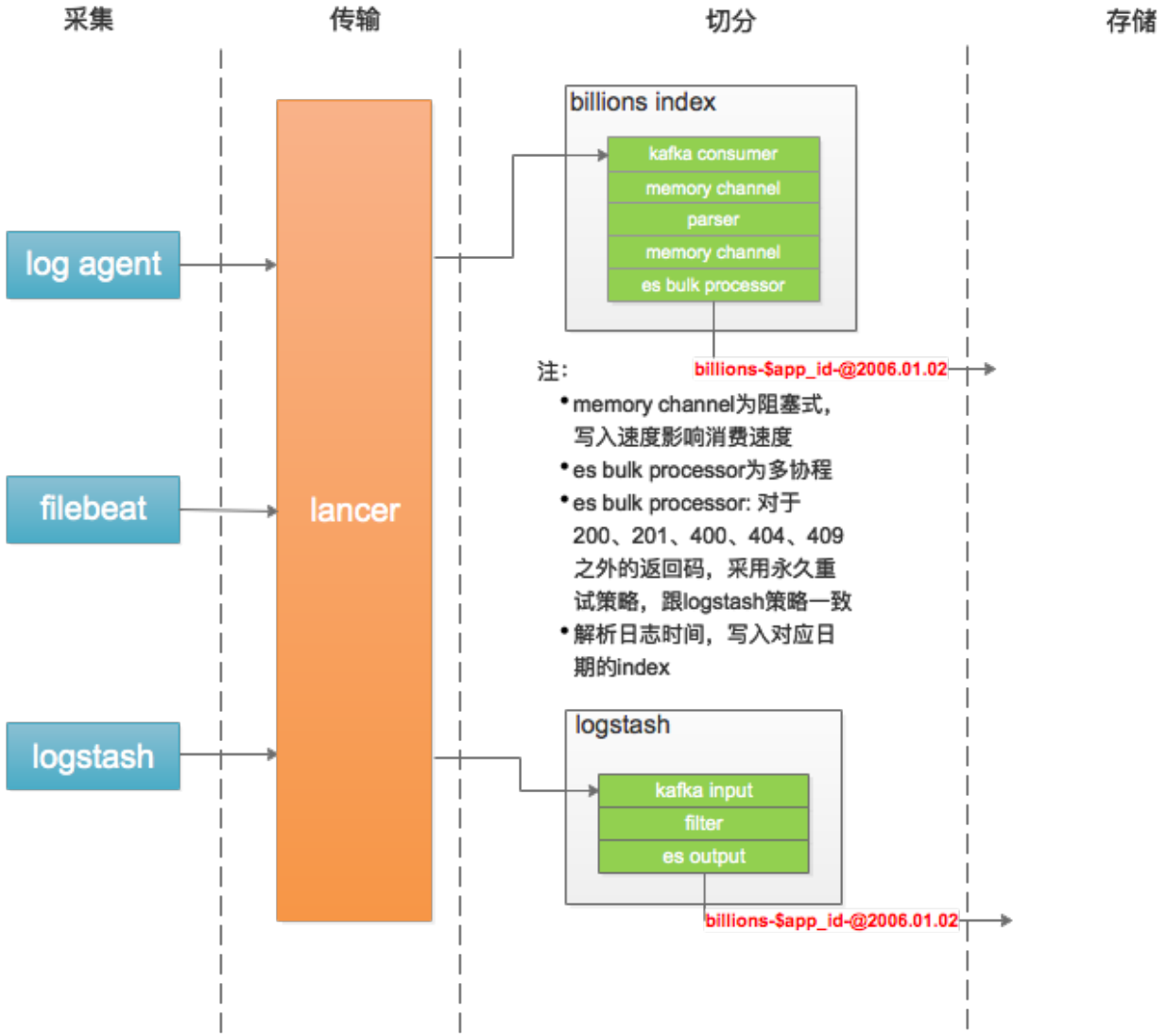
日志系统1.0-传输

- 传输
  - 基于公司内部统一的数据传输平台（lancer）



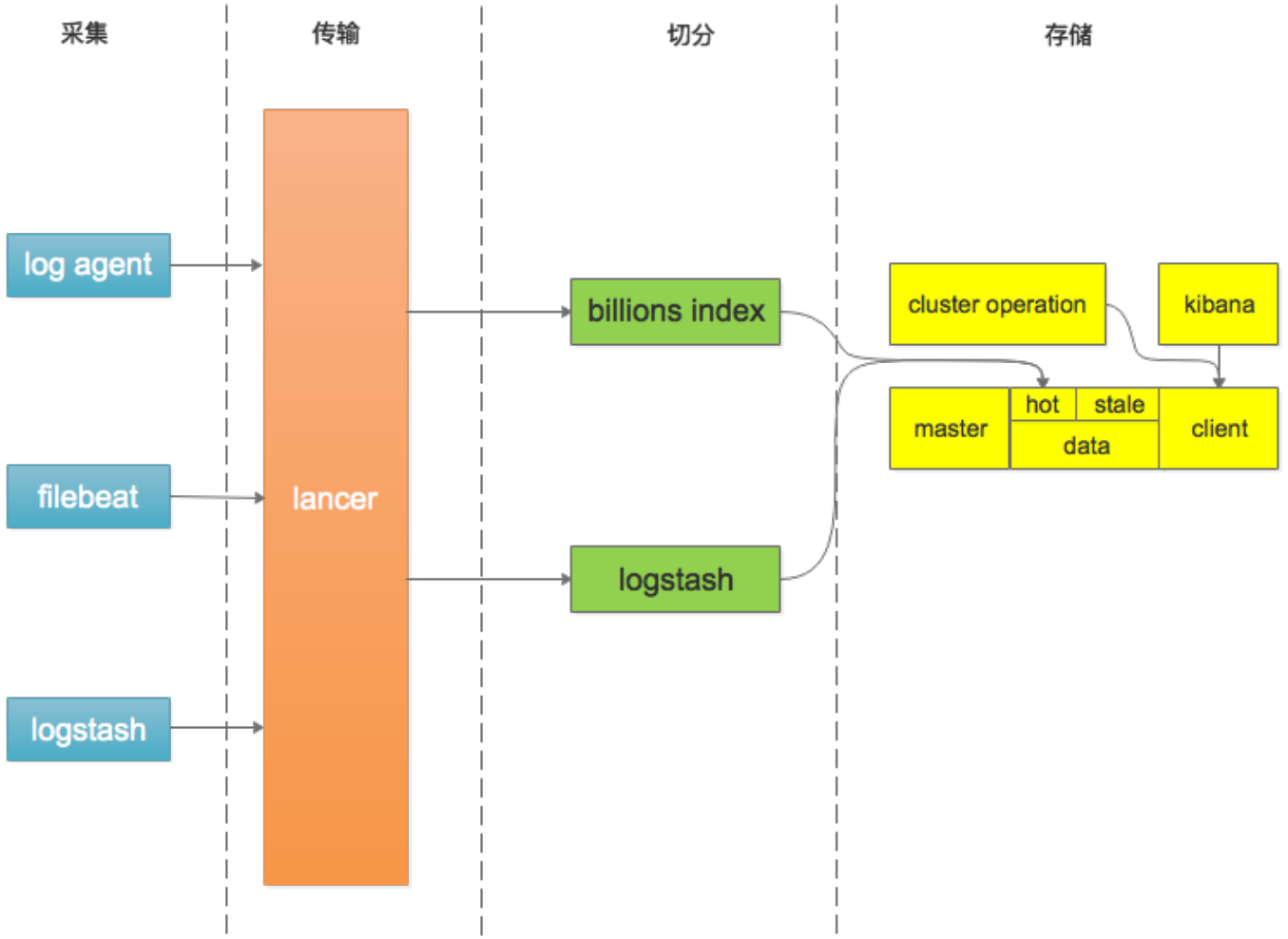
日志系统1.0-切分

- 切分
  - 从kafka消费日志，解析日志，写入elasticsearch。
  - 规则：billions-$app_id-@2006.01.02
  - 分为两种:
    - billions-index: 自研，golang开发，逻辑简单，性能高, 可定制化方便。
      - 日志规范产生的日志（log agent收集）
    - logstash：es官方组件，基于jruby开发，功能强大，资源消耗高，性能低。
      - 处理未按照日志规范产生的日志（filebeat、logstash收集），需配置各种日志解析规则。

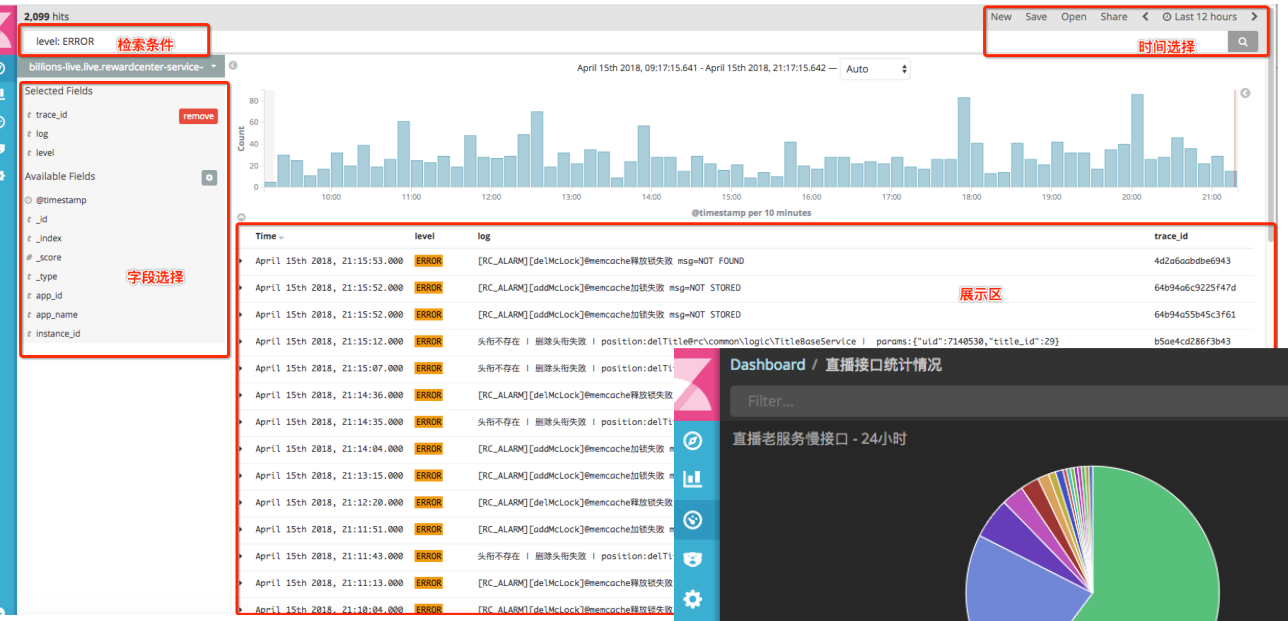


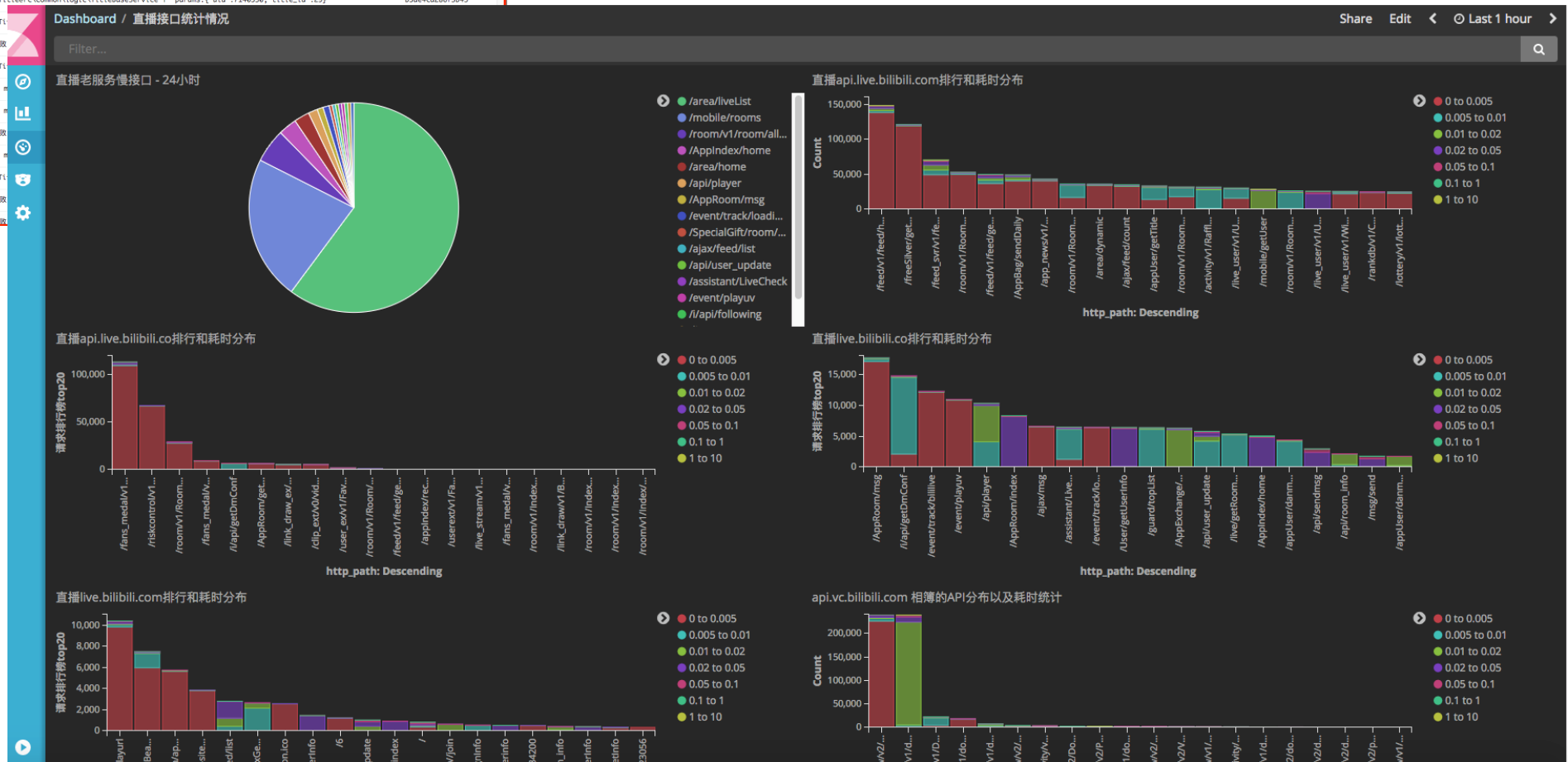
日志系统1.0-存储和检索

- 存储与检索
  - 存储基于elasticsearch
    - master node + data node(hot/stale) + client node
    - 每日固定时间进行热->冷迁移
    - Index提前一天创建，基于template进行mapping管理
  - 检索基于kibana



日志系统1.0-存储和检索



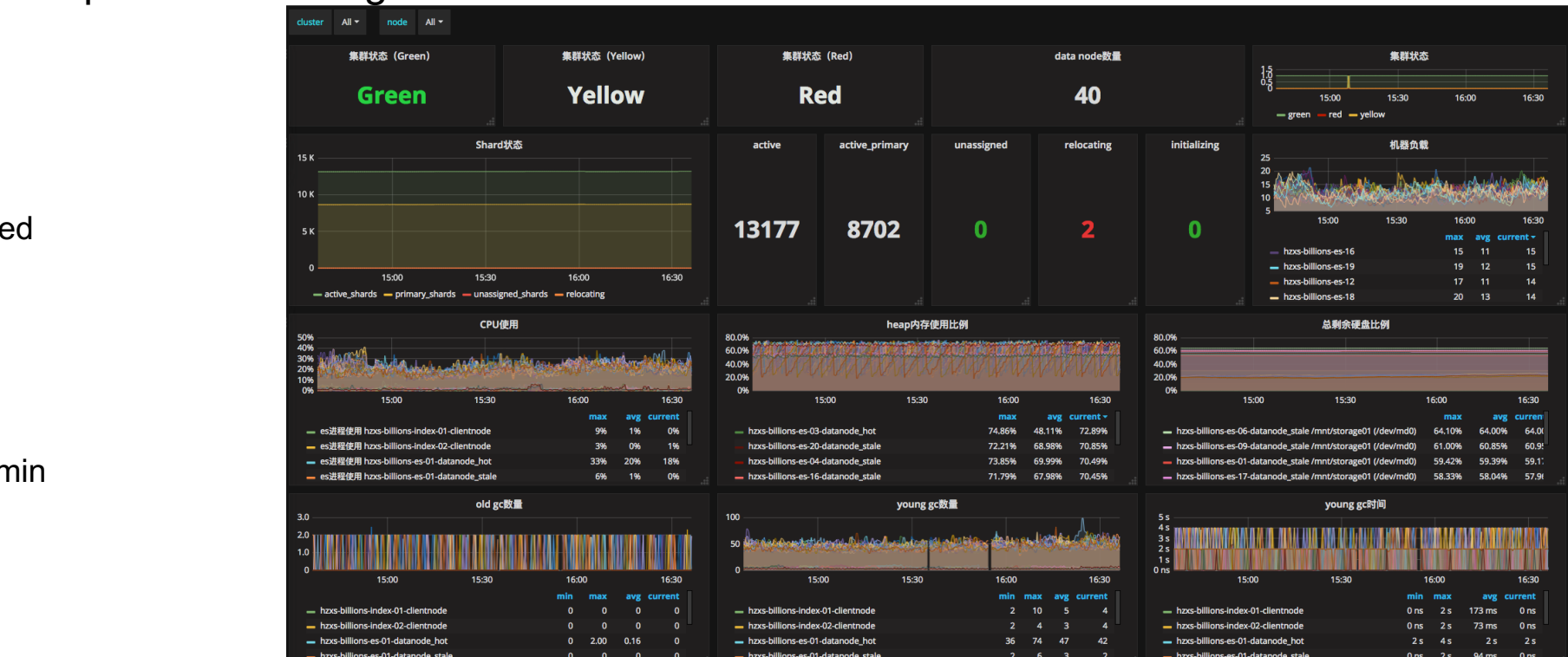


日志系统1.0精细化运维

日志系统重在运维！！

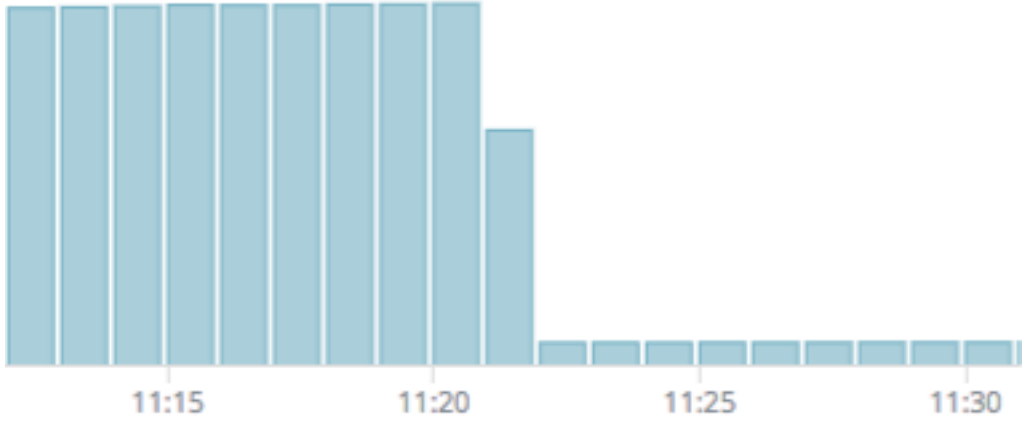
日志系统1.0运维-集群监控

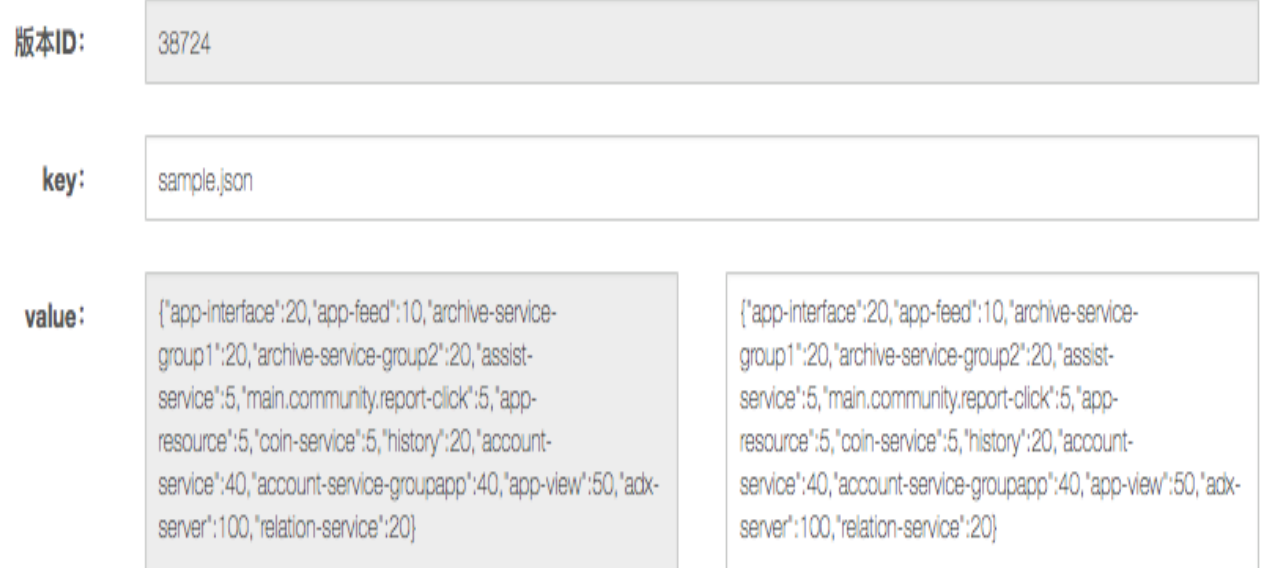
- 问题：如何了解Es集群的运行状态？
- 解决：Xpack + 自研监控 + es自身日志
  - 调用es api采集各类运行指标
  - es api -> es exporter -> prometheus -> grafana
  - 重点关注：
    - node num
    - shard num
    - heap usage/ GC
    - thread active/rejected
    - task in queue
    - node index rate
    - host load
  - 日志延迟
    - fake app：60 logs/min



日志系统1.0运维-日志采样

- 问题：如何控制业务日志量的大小？
- 解决：日志采样
  - 在log agent中增加了日志采样功能
  - 日志采样以app_id为维度，只对INFO级别以下日志进行随机采样
  - 采样率保存在配置中心，支持动态生效





日志系统1.0运维-shard管理

- 问题：
  - es index shard采用统一策略（5*2，5个primary，5个replica）
  - shard是有开销的（内存、文件句柄），数据量小的index没有必要创建5个shard
  - 某些index的单日数据量很大，导致单个shard过大，这样会导致检索的速度不是最优, 并且磁盘write IO集中在少数机器上。
- 解决：开发shard mng模块
  - 提前一天创建index
  - template一周更新一次
  - shard_num = max(avg(last 7 days) / 30GB , 2)
  - 集群shard数量降低了70%+，磁盘IO使用也更加高效

日志系统1.0运维-日志流监控

- 问题：
  - 日志是否延迟？是否丢失？
  - 哪一层出了问题（log agent? lancer? Index?）
- 解决： 日志流监控
  - 单一app维度下，根据日志产生时间，统计各层单位周期内的日志流量
  - 各层单位周期边界对齐
  - 目前实现了log agent和index层的监控
  - metric:

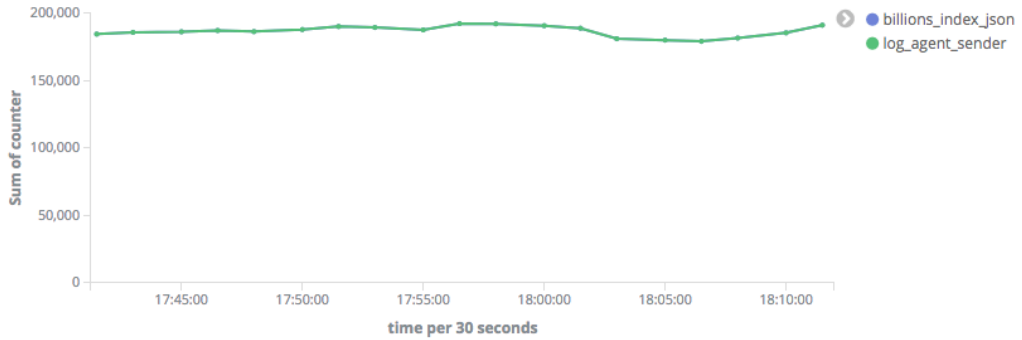
    {appId, timeRangeKey, source, status} counter


```
{
  "appId": "main.app-svr.stat-job",
  "timeRangeKey": "1529586500",
  "counter": 195,
  "source": "billions_index_json",
  "status": "received"
}
```

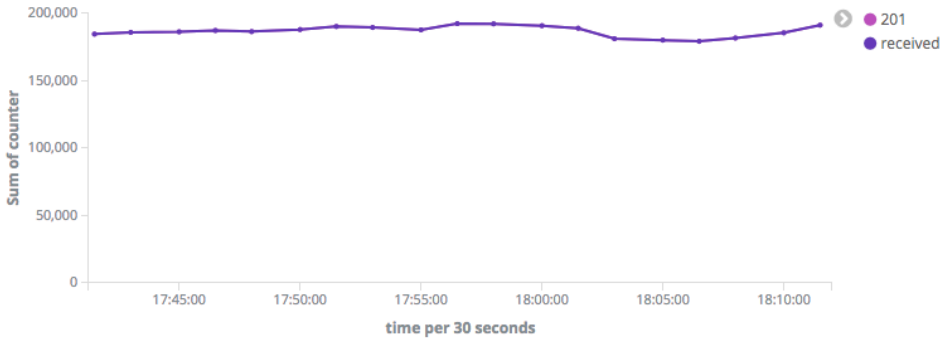
日志系统1.0运维-日志流监控

appId:"main.community.reply-job"

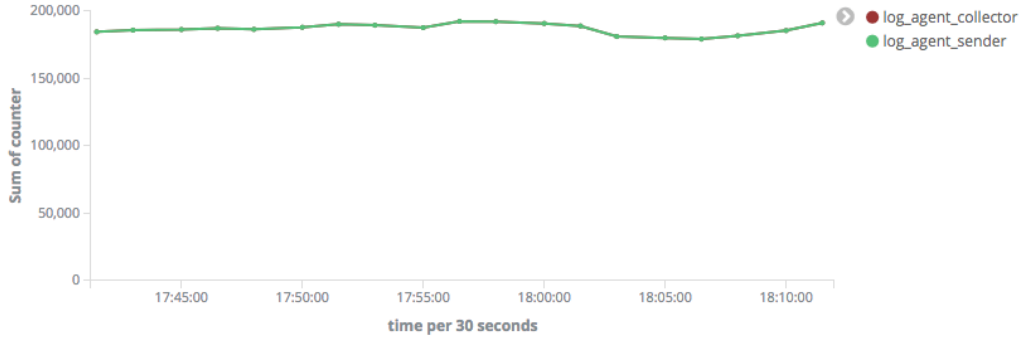
billions lancer写入与lancer读取 状态



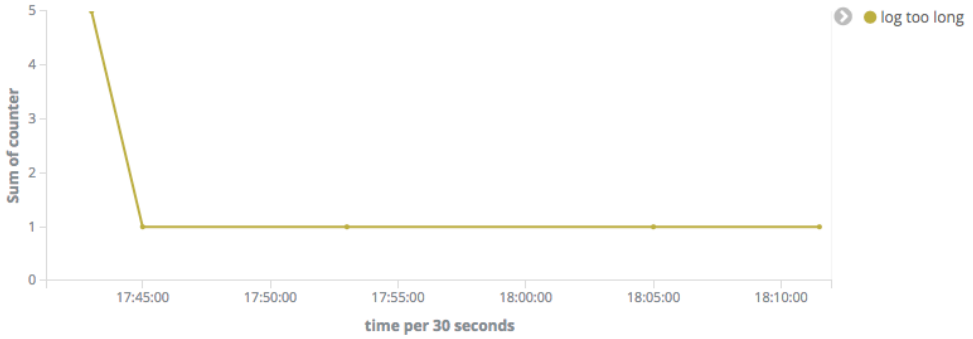
billions index lancer读取 与 接入es状态



billions log agent collector写入cache与sender写入lancer流量状态



billions log agent collector 非正常状态



billions log agent sender 非正常状态

😐

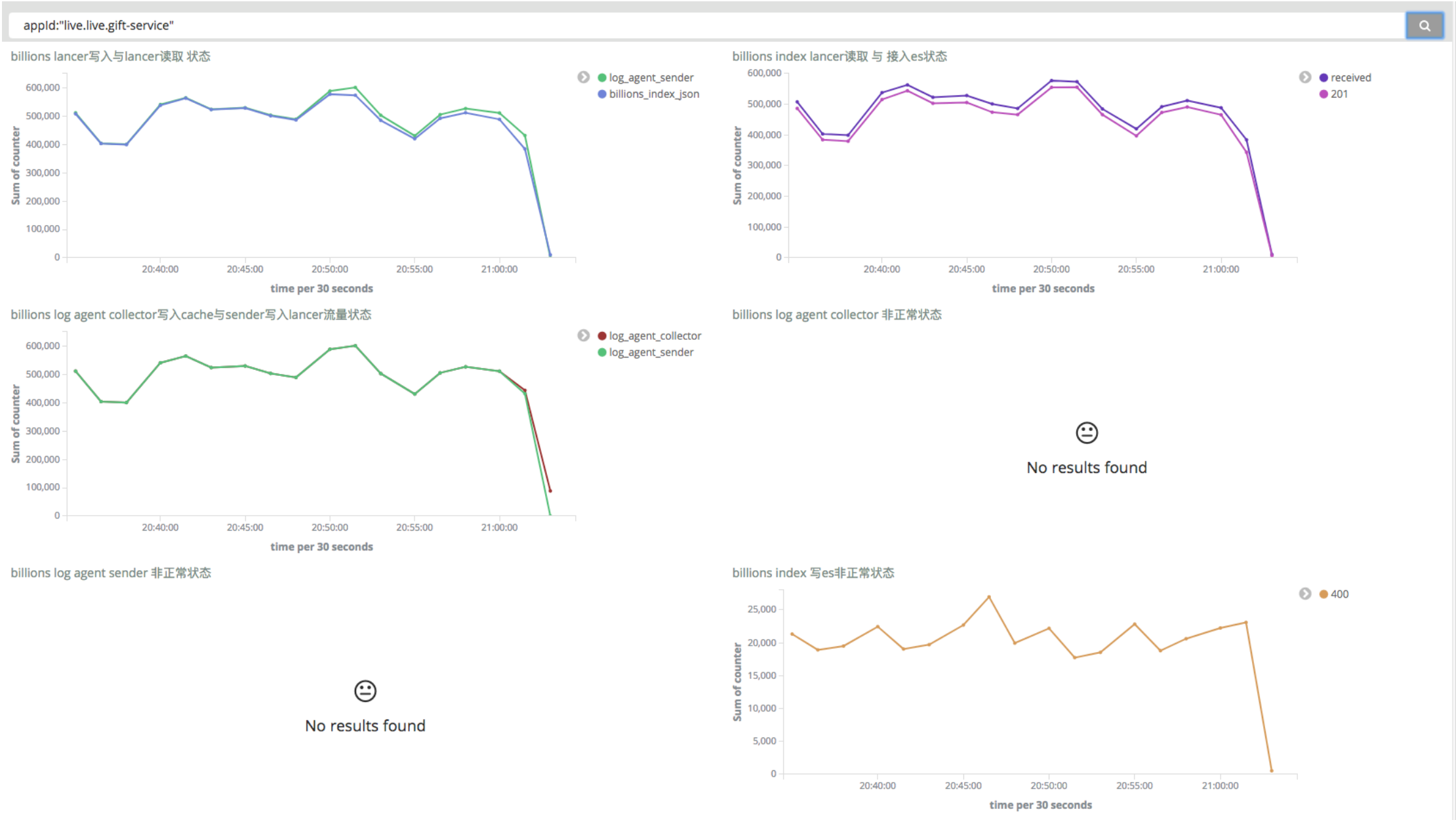
No results found

billions index 写es非正常状态

😐

No results found

日志系统1.0运维-日志流监控

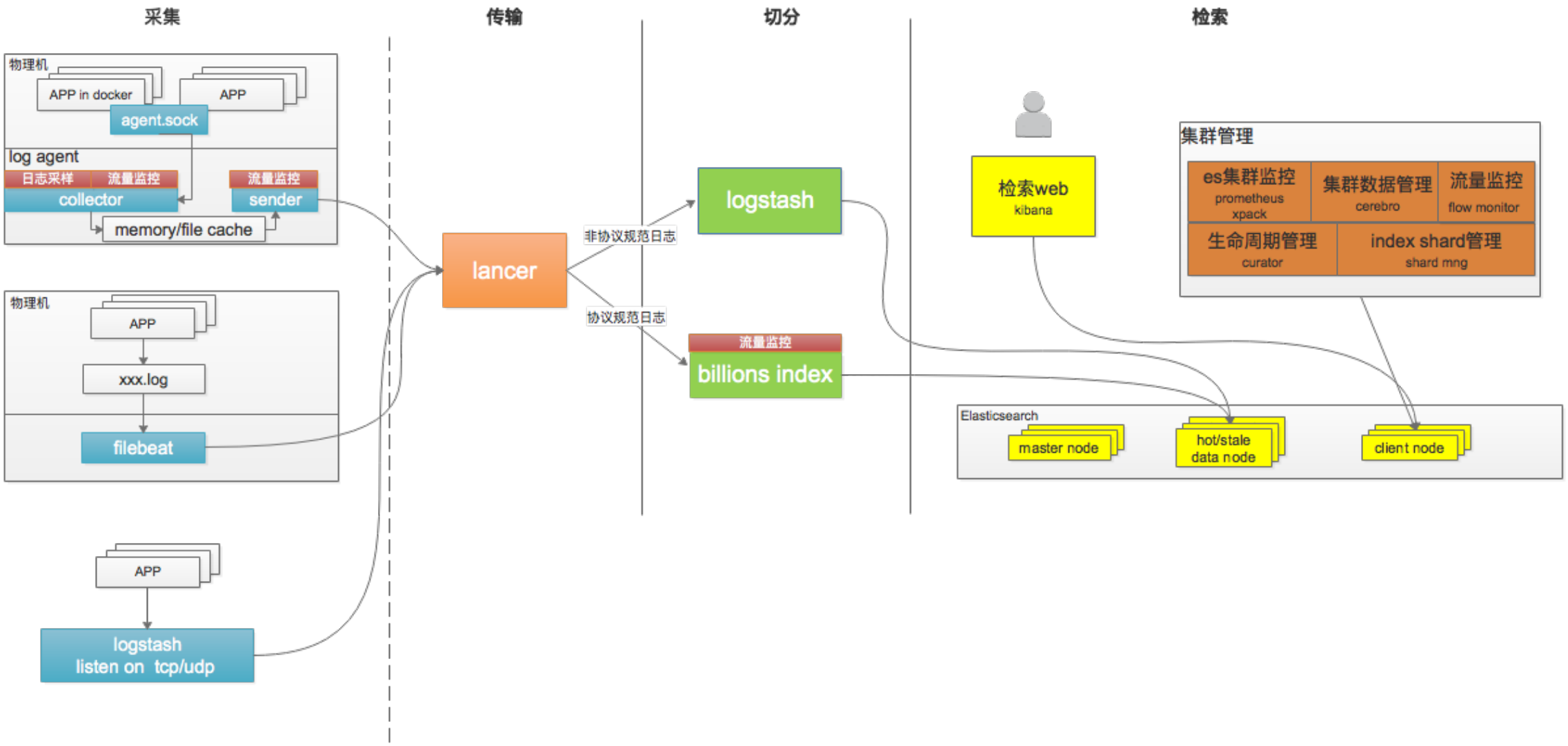


日志系统1.0运维-日志磁盘占用优化

- 问题：
  - 磁盘容量吃紧
- 解决方案：
  - index settings: _all.enabled: false
  - index dynamic_templates： only text for string fields
  - stale node
    - index.codec: best_compression
    - 定期对index force_merge
  - 效果：
    - 53.6% -> 20.9%，磁盘占用下降60%

编号	场景	数据占用大小	压缩比
1	原始数据（落盘文件）	3.6G	100%
2	按照es默认配置落入es	1.93GB	53.6%
3	2 之上 增加disable all（检索日志时，如未指明字段，需要遍历所有字段）	1.4GB	38.8%
4	3之上 优化mapping（减少不必要的索引字段）	1.1GB	30.5%
5	4之上 数据压缩方式调整为best_compression	771.6MB	20.9%

日志系统1.0 架构图


采集
传输
切分
检索
物理机
APP in docker
APP
agent.sock
log agent
日志采样
流量监控
流量监控
collector
sender
memory/file cache
物理机
APP
xxx.log
filebeat
APP
logstash
listen on tcp/udp
lancer
非协议规范日志
协议规范日志
logstash
流量监控
billions index
检索web
kibana
集群管理
es集群监控
prometheus
xpack
集群数据管理
cerebro
流量监控
flow monitor
生命周期管理
curator
index shard管理
shard mng
Elasticsearch
master node
hot/stale
data node
client node

日志系统1.0+（加强版）

- 功能诉求：
  - 基于日志实现报警
- 稳定性诉求

日志监控-技术选型

- X-Pack alert module
  - 功能全面，但是收费
  - 闭源，可定制化能力差
- Yelp/elastalert
  - 成熟度高（4K+star@github）
  - 监控种类丰富
  - 开源，方便二次开发
- 最终选定elastalert

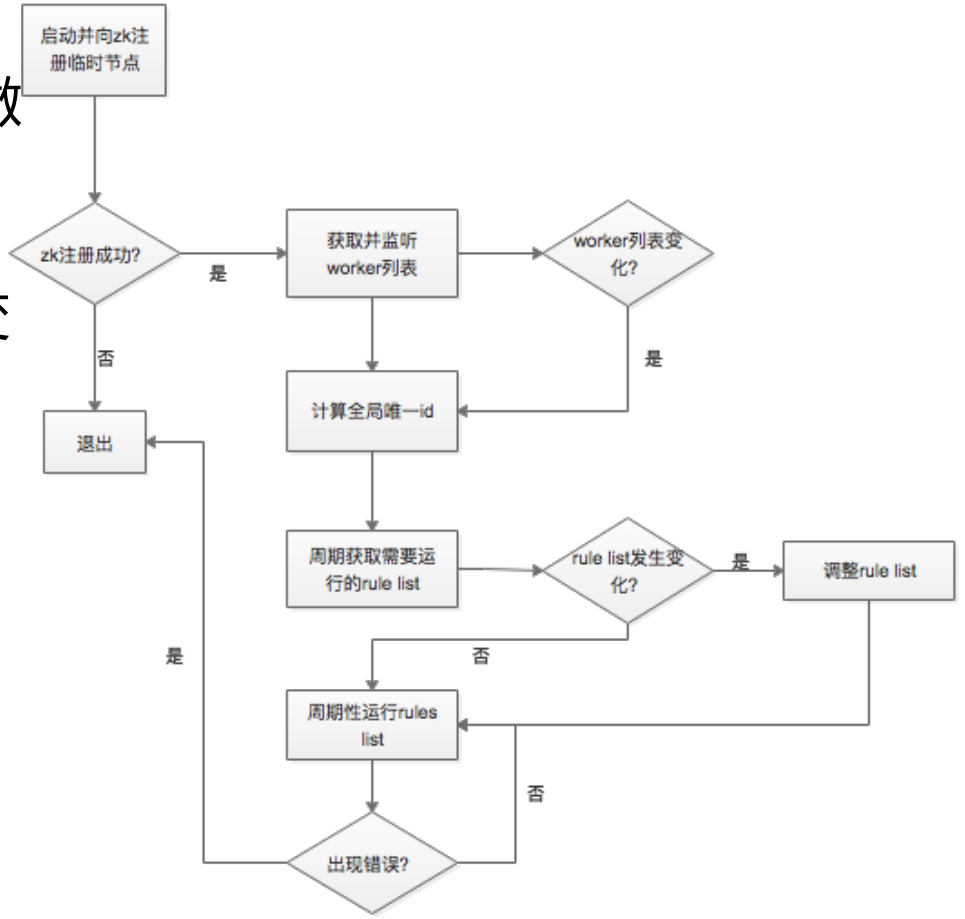
日志监控-elastalert问题

Elastalert自身还是存在一些问题亟待解决：

- 不支持分布式架构
  - 单实例架构，不能水平扩展，无法failover
  - Rules依赖配置文件存储，未入db
- 易用度不高
  - Rule配置项比较复杂
  - 不支持api配置rule
- rules串行执行，效率低

日志监控-elastalert二次开发

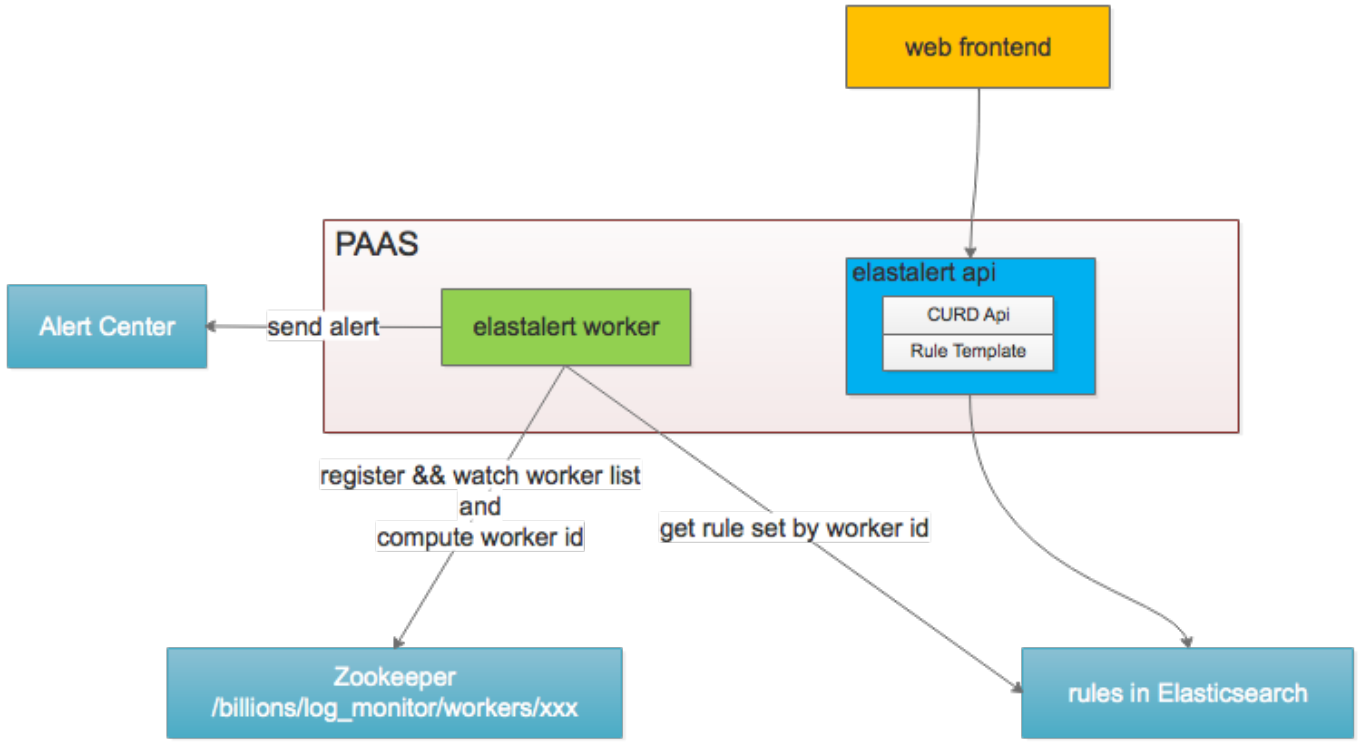
- rules存入elasticsearch中
  - 监控规则集中存储到数据库（elasticsearch）
  - 增加了rule存储的可靠性，也为elastalert的分布式实现做好准备
- 基于zookeeper实现分布式架构
  - 启动时以临时节点方式向zk注册，并监控worker list的变化
  - 计算全局唯一uniq_id
    - uniq_id = pos in sorted([worker...])
  - 进而获取rule list
    - rule list = [rule for rule in all_rules if hash(rule.id) % worker_count == uniq_id



日志监控-elastalert二次开发

- 降低配置复杂度
  - 所有类型的日志监控使用模板进行封装，以降低配置复杂度。
- Restful API
  - 封装出一套Restful api进行监控规则的增删改查
  - 开发web界面进行报警配置
- 与公司内部告警平台(skyeye)集成
- 报警消息格式调整（易读性、汉化）

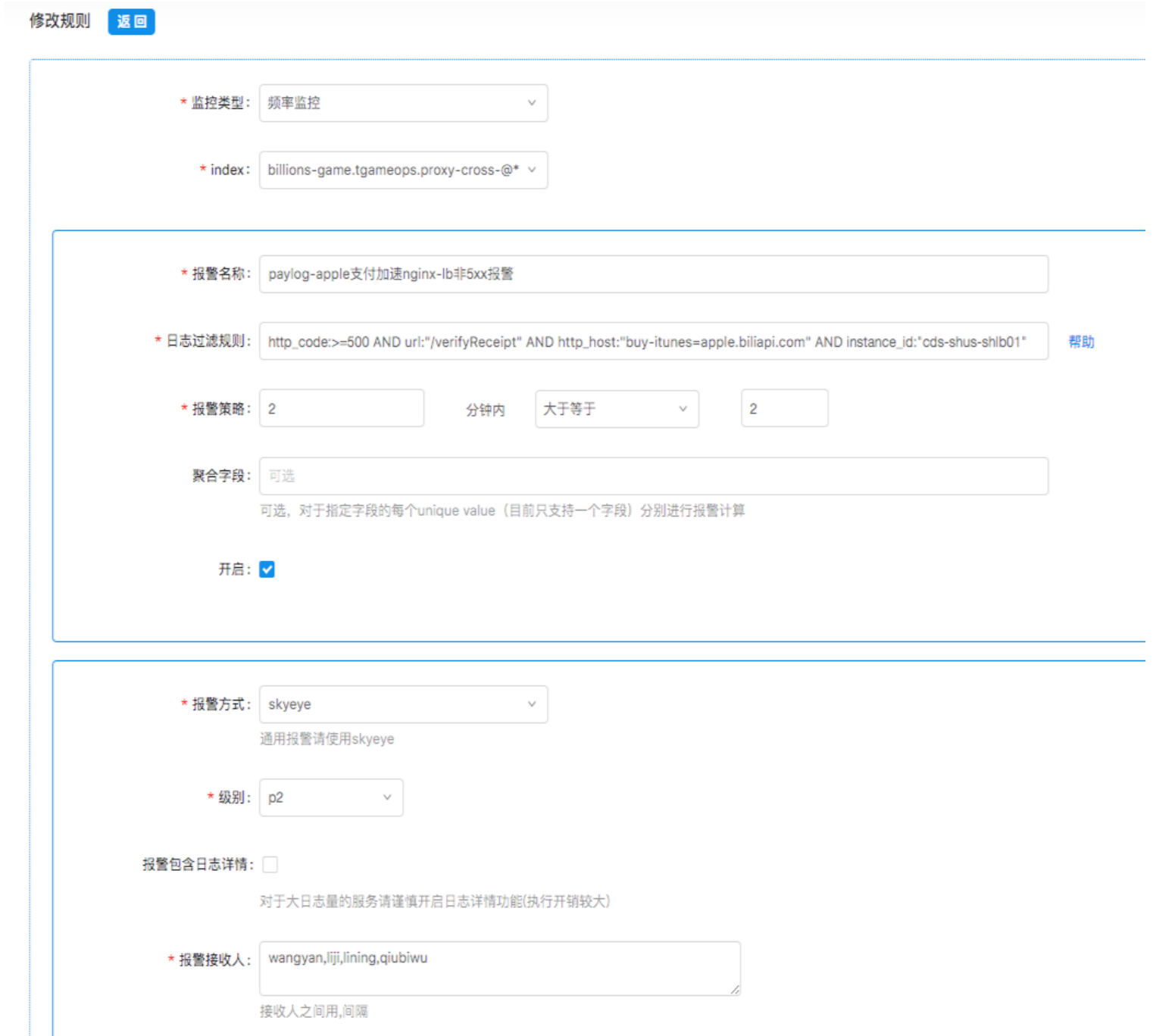
日志监控-架构


web frontend
PAAS
elastalert api
CURD Api
Rule Template
Alert Center
send alert
elastalert worker
register && watch worker list
and
compute worker id
get rule set by worker id
Zookeeper
/billions/log_monitor/workers/xxx
rules in Elasticsearch

日志监控-配置页面


请输入
添加规则
刷 新
状态
启用
告警规则
index
操作
mall-marketing WARN 报警
billions-mall-marketing-@*
修改
删除
[IM]msg_svr错误比例告警
billions-bplus-access-msg_svr-@*
修改
删除
docker-slb重试告警
billions-slb-error-@*
修改
删除
open.pay.open-bcoin WARN 报警
billions-openplatform-pay-bcoin-@*
修改
删除
paylog-apple支付加速nginx-lb非5xx报警
billions-game.tgameops.proxy-cross-@*
修改
删除
DB慢日志query_time大于1s并且小于10s监控
billions-db-mysql_slow_log-*
修改
删除

日志监控-配置页面


修改规则
返回
* 监控类型：
频率监控
* index：
billions-game.tgameops.proxy-cross-@*
* 报警名称：
paylog-apple支付加速nginx-lb非5xx报警
* 日志过滤规则：
http_code:>=500 AND url:"/verifyReceipt" AND http_host:"buy-itunes=apple.biliapi.com" AND instance_id:"cds-shus-shlb01"
帮助
* 报警策略：
2
分钟内
大于等于
2
聚合字段：
可选
可选，对于指定字段的每个unique value（目前只支持一个字段）分别进行报警计算
开启：
* 报警方式：
skyeye
通用报警请使用skyeye
* 级别：
p2
报警包含日志详情：
对于大日志量的服务请谨慎开启日志详情功能(执行开销较大)
* 报警接收人：
wangyan,liji,lining,qiubiwu
接收人之间用,间隔

日志监控-报警样例

```
日志报警: bpay ERROR报警

发生了12次事件, 时间:2018-03-21 08:05 CST至2018-03-21 08:06 CST

详情示例:
@version: 1
HOSTNAME: pay-task-02
host: 172.18.1.122
level: ERROR
level_value: 40000
logger_name: com.bilibili.common.helper.ExecuteHelper
message: [第三方支付订单状态查询异步重试]rechargeOrderNo=20180321080552913592041 8, channel=1, [第三方支付订单查询异步重试]调用接口发生异常, retry=3
stack_trace: java.lang.RuntimeException: [第三方支付订单查询异步重试]调用接口发生异常
        at com.bilibili.bpay.service.service.impl.PayRechargeOrderServiceImpl.triggerSyncSingleTradeQuery(PayRechargeOrderServiceImpl.java:121) ~[bpay-
svr-service-1.0.0-SNAPSHOT.jar:na]
        at com.bilibili.bpay.task.worker.TradeQueryWorker$1.executeWithoutResult(TradeQueryWorker.java:98) ~[TradeQueryWorker$1.class:na]
        at com.bilibili.common.helper.ExecuteHelper$ExecutableWithoutResult.execute(ExecuteHelper.java:142) ~[bpay-svr-common-1.0.0-SNAPSHOT.jar:na]
        at com.bilibili.common.helper.ExecuteHelper.execute(ExecuteHelper.java:47) ~[bpay-svr-common-1.0.0-SNAPSHOT.jar:na]
        at com.bilibili.bpay.task.worker.TradeQueryWorker.callBpayRechargeResultApi(TradeQueryWorker.java:91) [TradeQueryWorker.class:na]
        at com.bilibili.bpay.task.worker.TradeQueryWorker.execute(TradeQueryWorker.java:71) [TradeQueryWorker.class:na]
        at com.bilibili.bpay.task.consumer.RabbitConsumer.lambda$run$1(RabbitConsumer.java:45) [RabbitConsumer.class:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) ~[na:1.8.0_60]
        at java.util.concurrent.FutureTask.run(FutureTask.java:266) ~[na:1.8.0_60]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) ~[na:1.8.0_60]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) ~[na:1.8.0_60]
        at java.lang.Thread.run(Thread.java:745) ~[na:1.8.0_60]
Caused by: java.lang.RuntimeException: 调用bpay返回错误码:-113需重试
        at com.bilibili.bpay.persist.dao.impl.BiliAPIDaoImpl.syncTradeQuery(BiliAPIDaoImpl.java:129) ~[bpay-svr-persist-1.0.0-SNAPSHOT.jar:na]
        at com.bilibili.bpay.service.service.impl.PayRechargeOrderServiceImpl.triggerSyncSingleTradeQuery(PayRechargeOrderServiceImpl.java:118) ~[bpay-
svr-service-1.0.0-SNAPSHOT.jar:na]
        ... 11 common frames omitted

tag: bpay-svr-task
thread_name: pool-6-thread-1
```

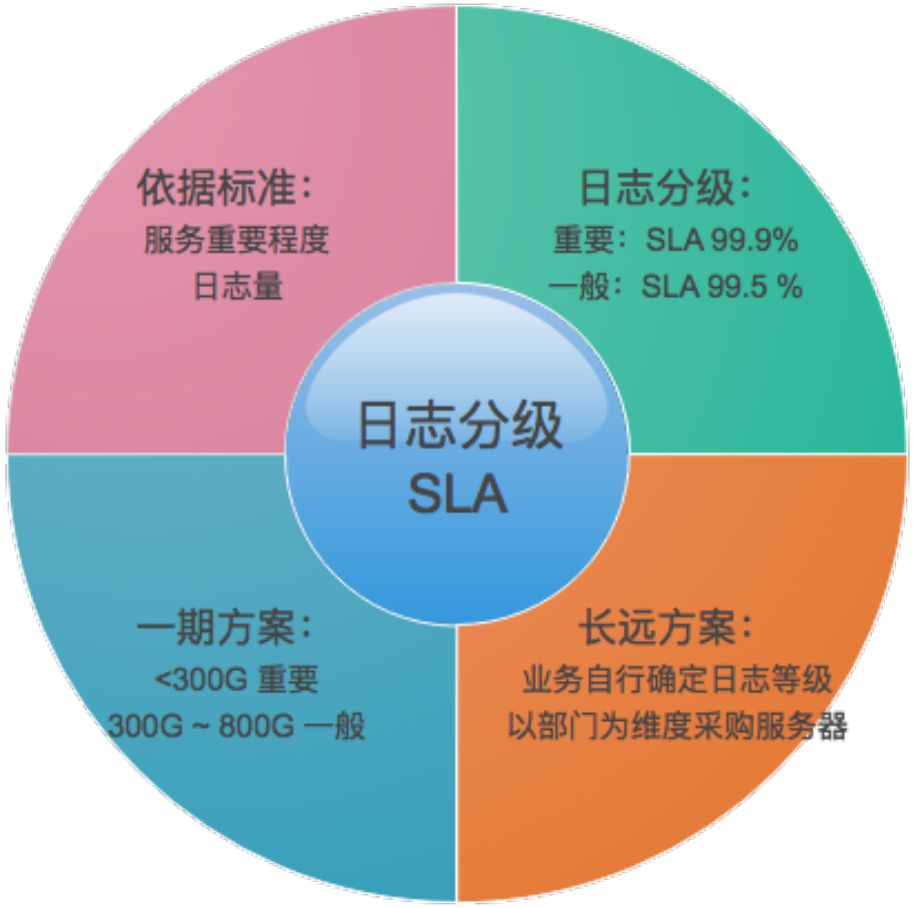
```
日志报警: data.bilibili.co/pegasus/feed/xx request_time>0.2s 比例报警

当http_path=/pegasus/feed/15时, 比例超出范围, 当前比例为: 29.53% (正常范围0%-10%), 时间:2018-03-20 22:01 CST至2018-03-20 22:11 CST
```

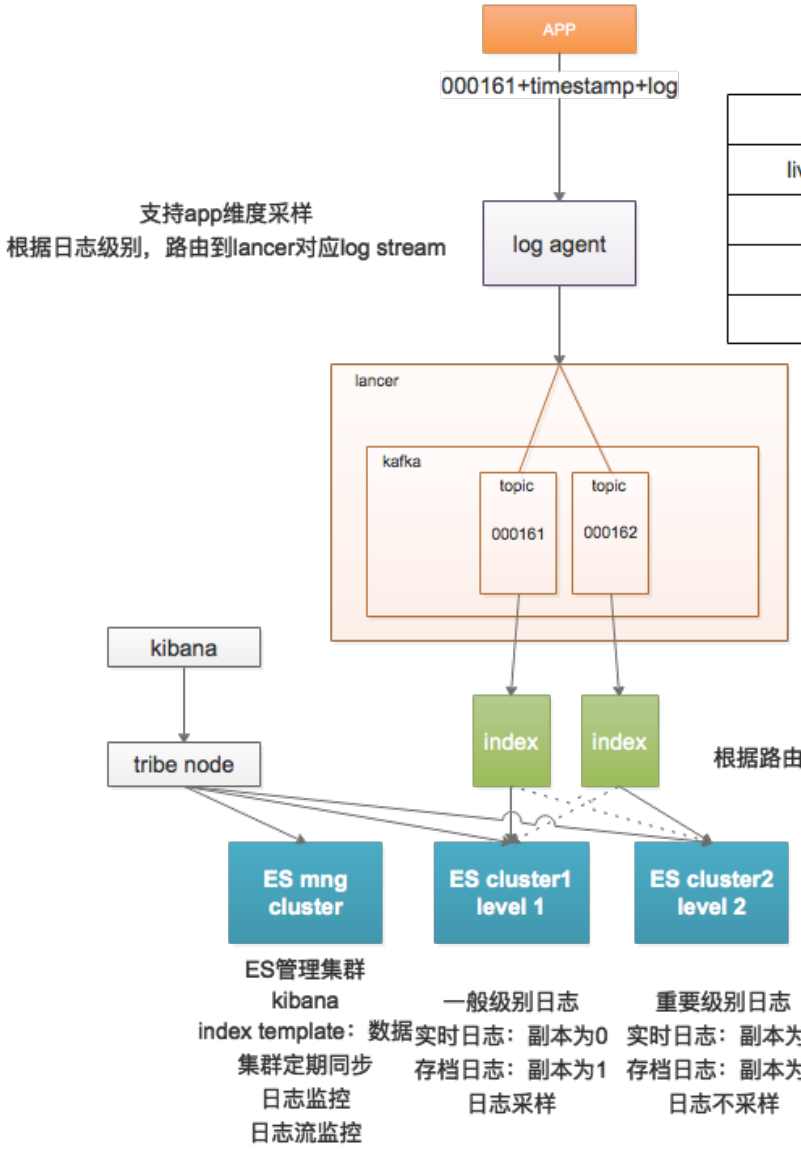
日志系统1.0+ -稳定性提升

- 不稳定诱因：
  - 单机群：一旦集群出现问题，影响所有服务日志。
  - 未制定SLA：所有日志按照相同服务等级对待，不论重要程度和日志量
- 应对方案：
  - 日志分级，制定SLA
  - 多集群
  - 制定故障预案
  - 兜底方案

日志分级和SLA


依据标准：
服务重要程度
日志量
日志分级：
重要：SLA 99.9%
一般：SLA 99.5 %
日志分级
SLA
一期方案：
<300G 重要
300G ~ 800G 一般
长远方案：
业务自行确定日志等级
以部门为维度采购服务器

多集群及故障预案



app_id	level
live.live.activity-service	1
live.live.relation	2

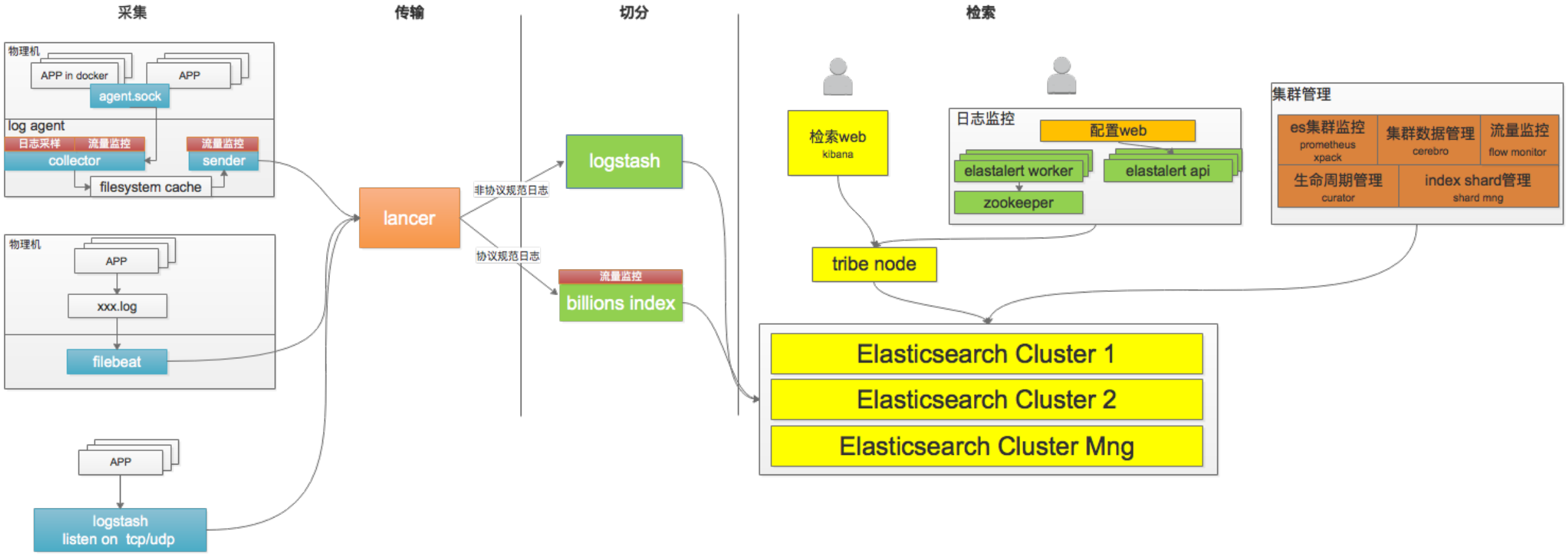
+

level	log_id
1	000161
2	000162

topic	es
topic_000161	cluster01
topic_000162	cluster02

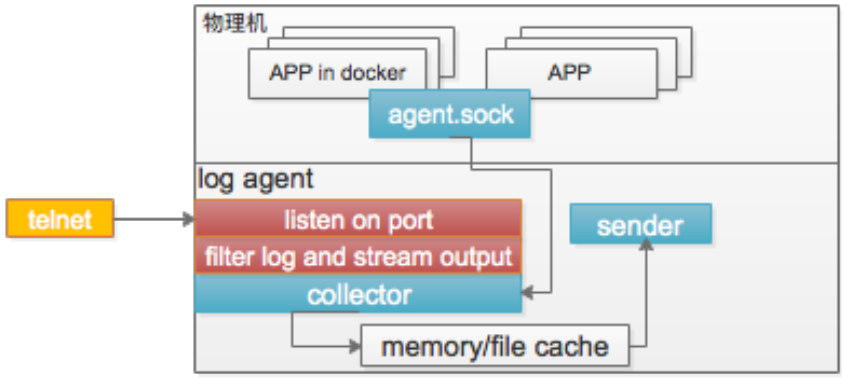
故障预案	
重要集群故障	(1) 重要日志切换到一般集群，不采样，副本为0 (2) 视一般集群负载，一般等级日志整体采样
一般集群故障	(1) 一般日志切换到重要集群，副本为0，采样率10%

日志系统1.0+ 架构图


采集
传输
切分
检索
物理机
APP in docker
APP
agent.sock
log agent
日志采样
流量监控
collector
流量监控
sender
filesystem cache
物理机
APP
xxx.log
filebeat
APP
logstash
listen on tcp/udp
lancer
非协议规范日志
协议规范日志
logstash
流量监控
billions index
检索web
kibana
日志监控
配置web
elastalert worker
elastalert api
zookeeper
集群管理
es集群监控
prometheus
xpack
集群数据管理
cerebro
流量监控
flow monitor
生命周期管理
curator
index shard管理
shard mng
tribe node
Elasticsearch Cluster 1
Elasticsearch Cluster 2
Elasticsearch Cluster Mng

兜底方案-日志本地查看

- 如果日志系统彻底不可用？？？？ 要让业务看到最新的日志
- 兜底方案：本地查看日志



兜底方案-日志本地查看

```
root@shylf-k8s-node-50:/data/lancer/collector/data # telnet 127.0.0.1 5555
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

 _        __ _____  _      ____   ___   __  __  _____
 \ \      / /|  ___|| |    / ___| / _ \ |  \/  ||  ___|
  \ \ /\ / / | |_   | |    | |    | | | || |\/| || |_
   \ V  V /  |  _|  | |    | |    | | | || |  | ||  _|
    \ /\ /   | |___ | |___ | |___ | |_| || |  | || |___
     V  V    |_____||_____| \____| \___/ |_|  |_||_____|

查询某一app日志请输入以下命令 get app_id
eg: get ops.billions.test

§ get main.app-svr.app-resource
00005315295842422220{"app_id":"main.app-svr.app-resource","caller":"","env":"prod","err":"","instance_id":"app-resource-13430-7585988578-7nnph","ip":"210.47.223
dd445e4f18ff5cdcf\u0026actionKey=appkey\u0026appkey=27eb53fc9058f8c3\u0026build=6790\u0026device=phone\u0026mobi_app=iphone\u0026platform=ios\u0026sign=017af42
ry/net/http/blademaster.Logger.func1:55","stack":"\u003cnil\u003e","time":"2018-06-21T20:30:42.2206","title":"/x/v2/notice","traceid":"8737898799968589385","ts

00005315295842422228{"app_id":"main.app-svr.app-resource","caller":"","env":"prod","err":"","instance_id":"app-resource-13430-7585988578-7nnph","ip":"183.198.19
ild=5271000\u0026channel=oppo\u0026mobi_app=android\u0026plat=0\u0026platform=android\u0026sign=0d940b9ab446fa4bd597c214fb8b69b2\u0026ts=1529584242\u0026type=0
nil\u003e","time":"2018-06-21T20:30:42.228467","title":"/x/v2/notice","traceid":"8549206342070341369","ts":0.000084016,"user":"no_user","zone":"sh001"}

00005315295842422234{"app_id":"main.app-svr.app-resource","caller":"","env":"prod","err":"","instance_id":"app-resource-13430-7585988578-7nnph","ip":"111.193.20
build=5271000\u0026mobi_app=android\u0026platform=android\u0026sign=b63bd3f5ba20414befece59ffb54683a\u0026ts=1529584242","path":"/x/resource/sidebar","ret":0,"
30:42.234296","title":"/x/resource/sidebar","traceid":"7355241632064734453","ts":0.000049583,"user":"no_user","zone":"sh001"}

00005315295842422239{"app_id":"main.app-svr.app-resource","caller":"","env":"prod","err":"","instance_id":"app-resource-13430-7585988578-7nnph","ip":"124.238.20
ml5_app_bili\u0026mobi_app=android\u0026model=vivo+X6S+A\u0026old_id=tBXA1v9EeQDWvS1YMZ%2FtYD%2FGNvI%3D\u0026sdkint=23\u0026seed=0","path":"/x/v2/version/updat
18-06-21T20:30:42.23917","title":"/x/v2/version/update","traceid":"3366829276821008277","ts":0.000094224,"user":"no_user","zone":"sh001"}

00005315295842422244{"app_id":"main.app-svr.app-resource","caller":"","env":"prod","err":"-304","instance_id":"app-resource-13430-7585988578-7nnph","ip":"36.249
3436\u0026build=5270000\u0026mobi_app=android\u0026platform=android\u0026sign=b0e739bcf592e43f62e350d5ec761902\u0026ts=1529584242\u0026ver=3701788055118495150"
```

日志系统2.0展望

- 1.0改进：
  - index和mapping管理：上报数据检查、mapping限制和自定义
  - 更加丰富和简易的日志上报方式： 例如直接捕获stdout
  - 资源隔离
- OLAP platform：
  - 消除日志数据孤岛，与公司大数据现有平台之间打通，功能互补
  - 基于es强大的数据检索和分析能力，深度挖掘数据的价值

QA