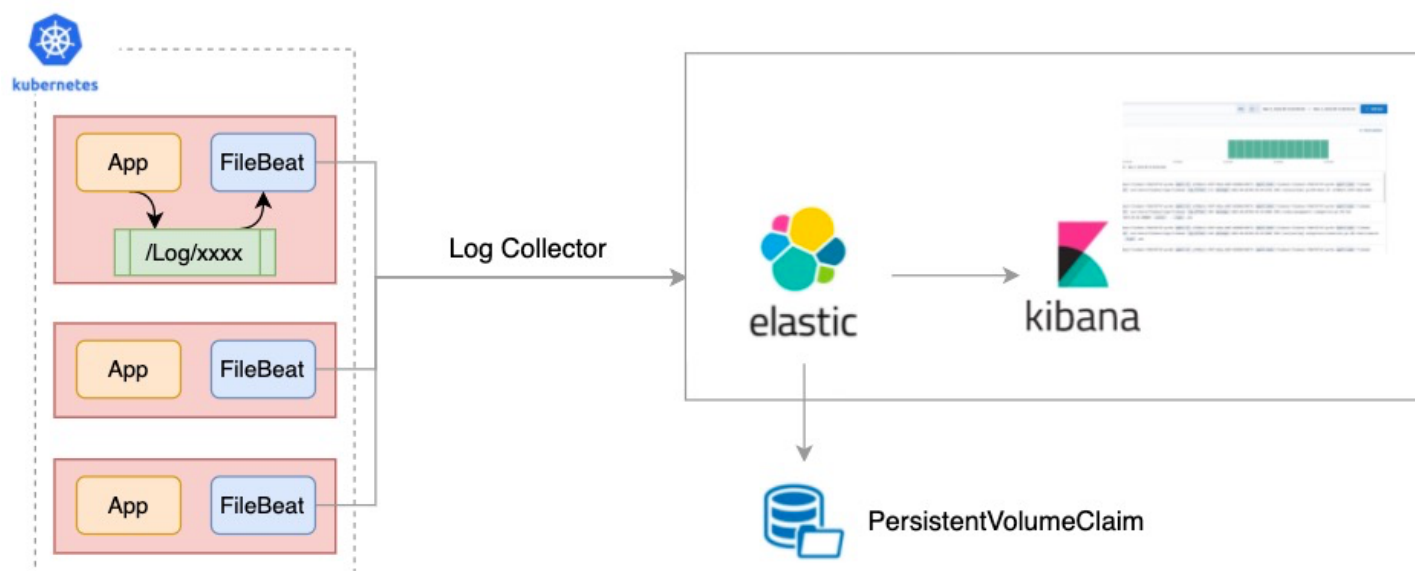


seecoder日志系统

1. 日志采集、检索

1.1 整体框架



1.2 具体设计

1. 日志格式示例

- a. 2025-12-01 21:30:45.123 [order-service] [main] INFO com.example.OrderService - createOrder|0b0f945316660590407391087e36cf|0.1.1.1|mobile|vip

1.2.1 traceId与spanId埋点

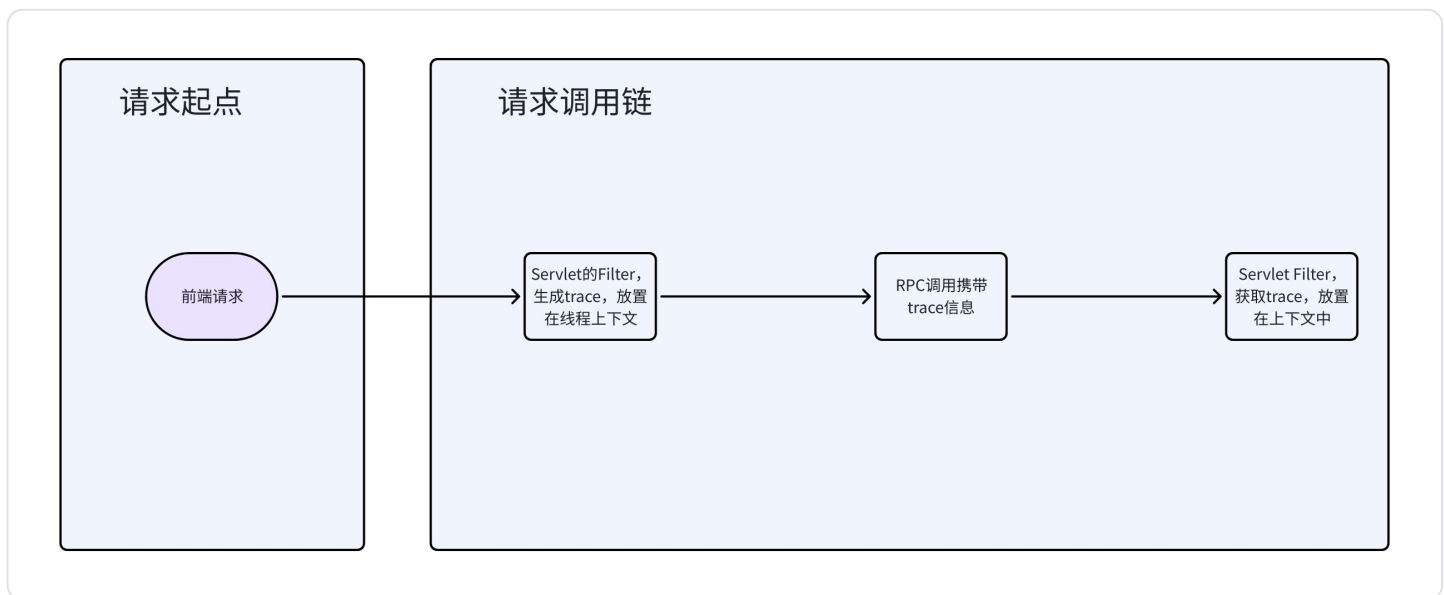
1.2.1.1 traceId

作用：追踪一次请求



生成方式：

1. 请求开始基于Servlet的Filter的机制，接受前端请求时生成traceId，放置在线程上下文中
2. 随Rpc调用传递

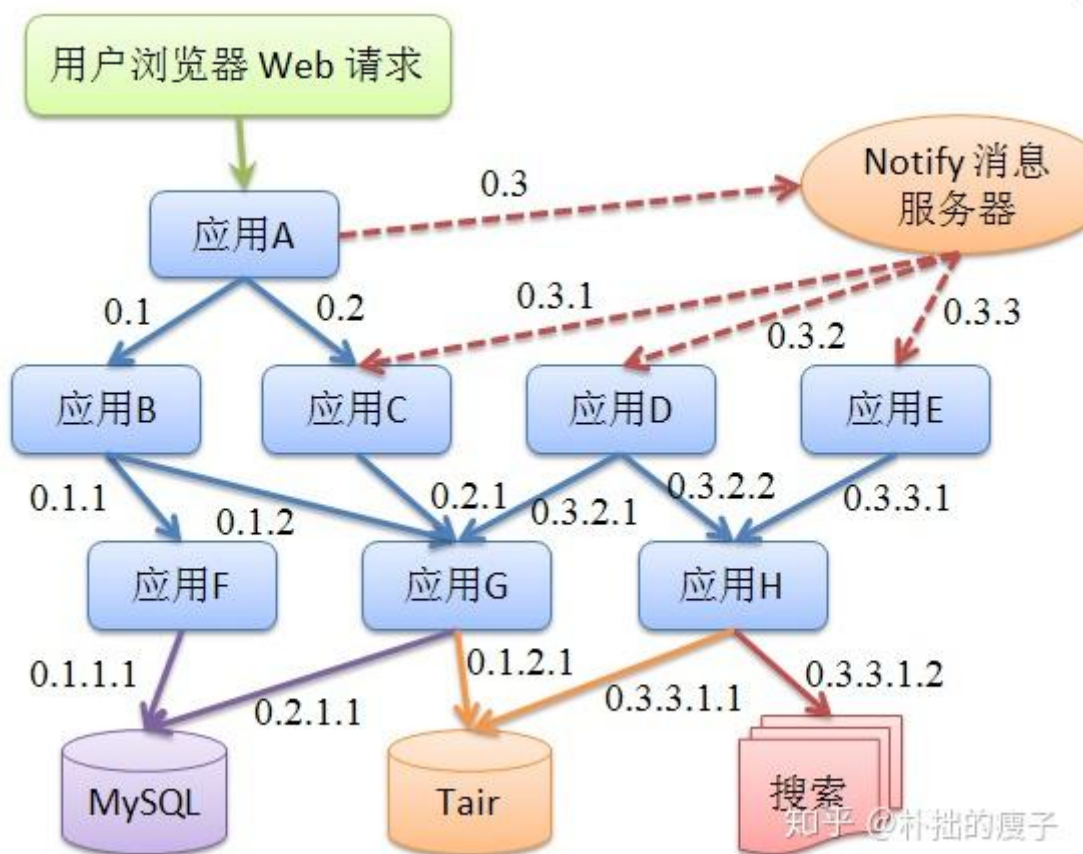


1.2.1.2 spanId

作用：用于区分同一个调用链下的多个网络调用的发生顺序和嵌套层次关系

生成规则：

1. 同级Rpc调用，计数器+1
2. 跨层级调用，深度+1



1.2.2 日志采集、检索

1.2.2.1 EFK体系介绍

EFK (Elasticsearch + Fluentd/Fluent Bit + Kibana) 是一种广泛用于日志采集、存储、分析和可视化的开源技术栈。它常被部署在 Kubernetes 环境或分布式系统中，用于集中管理日志数据。

1. Elasticsearch

- 分布式搜索与分析引擎，用于存储和索引日志数据。
- 支持全文检索、聚合分析、高可用和水平扩展。

2. Fluentd / Fluent Bit

- 日志收集器 (Log Collector)，负责从各种来源（如容器 stdout、文件、syslog 等）采集日志。
- Fluentd 功能强大但资源占用较高；Fluent Bit 是轻量级版本，更适合边缘设备或资源受限环境（如 Kubernetes 节点上的 DaemonSet）。

3. Kibana

- 数据可视化工具，提供 Web 界面用于查询、过滤、图表展示 Elasticsearch 中的日志。

1.2.2.2 详细设计

1.2.2.2.1 基于Fluent Bit的日志采集

1. Fluent bit部署

使用daemon set在每个节点上均部署一个

代码块

```
1  # fluent-bit-ds.yaml
2  apiVersion: apps/v1
3  kind: DaemonSet
4  metadata:
5    name: fluent-bit
6    namespace: logging
7    labels:
8      app: fluent-bit
9  spec:
10   selector:
11     matchLabels:
12       app: fluent-bit
13   template:
14     metadata:
15       labels:
16         app: fluent-bit
17     spec:
18       serviceAccountName: fluent-bit
19       containers:
20       - name: fluent-bit
21         image: cr.fluentbit.io/fluent/fluent-bit:3.1.7
22         imagePullPolicy: Always
23         ports:
24         - containerPort: 2020
25           protocol: TCP
26         env:
27         - name: ES_HOST
28           value: "elasticsearch.logging.svc.cluster.local"
29         - name: ES_PORT
30           value: "9200"
31         volumeMounts:
32         - name: config
33           mountPath: /fluent-bit/etc
34         - name: logdir
35           mountPath: /var/log/myapp # 应用日志目录
36         - name: fluentbit-db
37           mountPath: /var/log
38       volumes:
39       - name: config
40         configMap:
41           name: fluent-bit-config
42       - name: logdir
43         hostPath:
```

```

44         path: /var/log/myapp                    # 宿主机日志路径
45     - name: fluentbit-db
46       hostPath:
47         path: /var/lib/fluent-bit
48         type: DirectoryOrCreate
49     tolerations:
50     - key: node-role.kubernetes.io/control-plane
51       operator: Exists
52       effect: NoSchedule
53     securityContext:
54       runAsUser: 0

```

2. Fluent bet 配置

代码块

```

1  # fluent-bit-config.yaml
2  apiVersion: v1
3  kind: ConfigMap
4  metadata:
5    name: fluent-bit-config
6    namespace: logging
7  data:
8    fluent-bit.conf: |
9      [SERVICE]
10         Flush            1
11         Log_Level        info
12         Daemon           off
13         Parsers_File     parsers.conf
14         HTTP_Server       On
15         HTTP_Listen      0.0.0.0
16         HTTP_Port        2020
17
18      [INPUT]
19         Name              tail
20         Path              /var/log/myapp/*.log
21         Parser            app_log_parser
22         Tag               kube.myapp.*
23         Refresh_Interval  10
24         Mem_Buf_Limit     5MB
25         Skip_Long_Lines   On
26         DB                /var/log/flb_myapp.db
27
28      [FILTER]
29         Name              lua
30         Match              kube.myapp.*

```

```

31      script  parse_message.lua
32      call    split_message
33
34      [FILTER]
35          Name      nest
36          Match     kube.myapp.*
37          Operation lift
38          Nested_under parsed_fields
39
40      [OUTPUT]
41          Name      es
42          Match     kube.myapp.*
43          Host      ${ES_HOST}
44          Port      ${ES_PORT}
45          Index     app-logs
46          Type      _doc
47          Time_Key   @timestamp
48          Include_Tag_Key On
49          Tag_Key    fluentbit_tag
50          Logstash_Format On
51          Replace_Dots On
52          Retry_Limit False
53
54      parsers.conf: |
55          [PARSER]
56              Name      app_log_parser
57              Format     regex
58              Regex     ^(<time>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}\.\d{3}) \[(?
<app_name>[^\]]+)\] \[(?<thread>[^\]]+)\] (?<level>\w+)\s+(?<logger>[^\ ]+) - (?
<message>.*)$
59              Time_Key  time
60              Time_Format %Y-%m-%d %H:%M:%S.%L
61              Time_Keep Off
62
63      parse_message.lua: |
64          function split_message(tag, timestamp, record)
65              local message = record["message"]
66              if not message then
67                  return 0, timestamp, record
68              end
69
70              local parts = {}
71              for part in string.gmatch(message, "([^\]]*)") do
72                  table.insert(parts, part)
73              end
74
75              local parsed = {}

```

```
76     if #parts >= 1 then parsed["biz_name"] = parts[1] end
77     if #parts >= 2 then parsed["trace_id"] = parts[2] end
78     if #parts >= 3 then parsed["rpc_id"]    = parts[3] end
79
80     local context = {}
81     for i = 4, #parts do
82         table.insert(context, parts[i])
83     end
84     parsed["context"] = context
85
86     record["parsed_fields"] = parsed
87     return 2, timestamp, record
88 end
```

3. Elastic search存储示例

代码块

```
1  {
2    "@timestamp": "2025-12-20T21:30:45.123Z",
3    "app_name": "order-service",
4    "thread": "main",
5    "level": "INFO",
6    "logger": "com.example.OrderService",
7    "biz_name": "createOrder",
8    "trace_id": "abc123",
9    "rpc_id": "rpc-456",
10   "context": ["user123", "mobile", "vip"],
11   "fluentbit_tag": "app.log"
12 }
```