|  |  |  |  |
| --- | --- | --- | --- |
| **修订历史** | | | |
| 日期 | 版本 | 说明 | 作者 |
| 7/13/2018 | 1.0 | 第一次迭代测试 | 金瑞洋、王见思、宋逸凡、李翌珺 |
| 7/26/2018 | 2.0 | 第二次迭代测试 |
| 9/4/2018 | 3.0 | 第三次及第四次迭代 |
|  |  |  |

**“慧眼识踪“——基于深度学习的人员即时搜寻系统**

**架构设计文档**

版本3.0

目录

[1.引言 3](#_Toc524294783)

[1.1 文档目的 3](#_Toc524294784)

[1.2 名词术语 3](#_Toc524294785)

[2. 需求概述 4](#_Toc524294786)

[2.1. 功能性 4](#_Toc524294787)

[2.1.1 Use case图及说明 4](#_Toc524294788)

[2.1.2“查看摄像头参数”用例规约 6](#_Toc524294789)

[2.1.3 查看实时监控”用例规约 7](#_Toc524294790)

[2.1.4“查看历史监控视频”用例规约 7](#_Toc524294791)

[2.1.5“根据图片搜索目标”用例规约 8](#_Toc524294792)

[2.2. 易用性 8](#_Toc524294793)

[2.3. 可靠性 9](#_Toc524294794)

[2.4. 性能 9](#_Toc524294795)

[2.5 可支持性 9](#_Toc524294796)

[3.架构设计思想 10](#_Toc524294797)

[4. 架构体系描述 10](#_Toc524294798)

[5. 架构图 11](#_Toc524294799)

[5.1部署视图 11](#_Toc524294800)

[5.2逻辑视图 12](#_Toc524294801)

[5.2.1 概述 12](#_Toc524294802)

[5.2.2具有重要意义的设计包 13](#_Toc524294803)

[5.3 实现视图 13](#_Toc524294804)

[5.4 运维监控架构 14](#_Toc524294805)

[5.5 日志管理架构 14](#_Toc524294806)

[6.技术选型 15](#_Toc524294807)

[7.过程视图 16](#_Toc524294808)

[7.1 通信协议 16](#_Toc524294809)

[7.2 时序图 16](#_Toc524294810)

[7.3 进程视图 17](#_Toc524294811)

[8.数据视图 18](#_Toc524294812)

[9 接口设计 18](#_Toc524294813)

[9.1 动态资源服务接口 18](#_Toc524294814)

[9.2 视频直播服务 19](#_Toc524294815)

[9.2.1 视频推流 19](#_Toc524294816)

[9.2.2 视频播流 19](#_Toc524294817)

[9.3 静态资源服务 20](#_Toc524294818)

[9.3.1 静态网页 20](#_Toc524294819)

[9.3.2 视频文件 20](#_Toc524294820)

[9.4 识别服务 20](#_Toc524294821)

[9.4.1 从直播中进行搜索 20](#_Toc524294822)

[9.4.2 从历史视频中进行搜索 20](#_Toc524294823)

[10 部署运维 21](#_Toc524294824)

[10.1集群管理 22](#_Toc524294825)

[10.1.1 主机节点 22](#_Toc524294826)

[10.1.2 镜像分布 23](#_Toc524294827)

[10.2 CI/CD 23](#_Toc524294828)

[10.3 API 管理 24](#_Toc524294829)

[10.4 数据监控与通知 24](#_Toc524294830)

[10.4.1 Prometheus 24](#_Toc524294831)

[10.4.2 Prometheus data exporter 25](#_Toc524294832)

[10.4.3 Grafana 25](#_Toc524294833)

[10.4.4 Alert Manager 26](#_Toc524294834)

[10.5 日志管理系统 27](#_Toc524294835)

[10.5.1 Fluentd 27](#_Toc524294836)

[10.5.1 Elasticsearch 27](#_Toc524294837)

[10.5.2 Kibana 27](#_Toc524294838)

[11 遗留问题 28](#_Toc524294839)

[12 项目成果 29](#_Toc524294840)

# 1.引言

本项目针对寻找走失人员、跟踪犯罪嫌疑人和定位传染病人等众多现实场景下都必须要解决的搜寻指定人员的问题，设计并开发了一个“基于深度学习的人员即时搜寻系统”。

## 1.1 文档目的

本文档是慧眼识踪（GETS）系统进行详细设计和编码的重要依据。对于整体系统结构关系进行详细描述，阐述系统总体框架，说明了系统中所采取的相关技术，并对于相关内容作出一定规定。为开发人员今后设计、编码、测试提供了可参考模板。

## 1.2 名词术语

搜索目标：需要搜寻系统根据监控画面锁定位置的人物

GETS：God Eye Tracking System， “慧眼识踪”人员搜寻系统

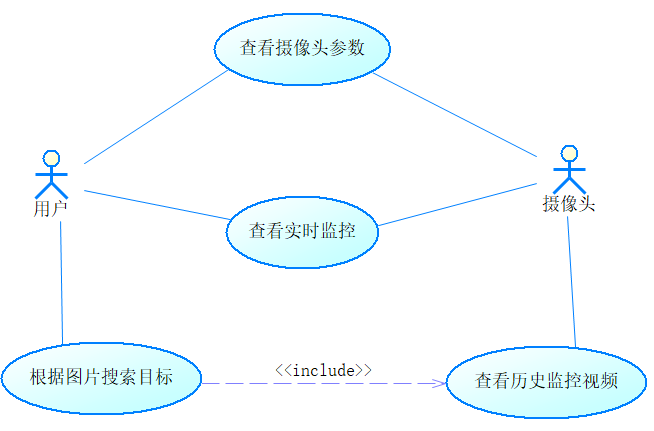
MTBF：Mean Time Between Failure，平均失效间隔时间

MTTR：Mean Time To Restoration，平均恢复时间

# 2. 需求概述

## 2.1. 功能性

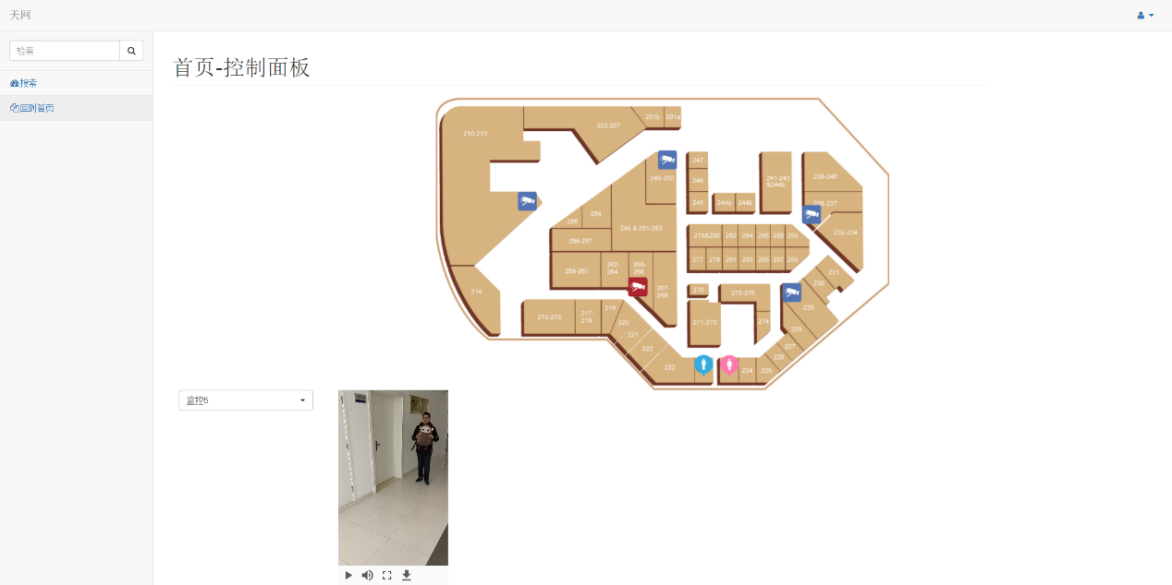
### 2.1.1 Use case图及说明



查看摄像头参数：允许用户查看某建筑物平面图，并查看摄像头位置以及相关参数（方向、俯角等），同时平面图自动显示摄像头覆盖位置。

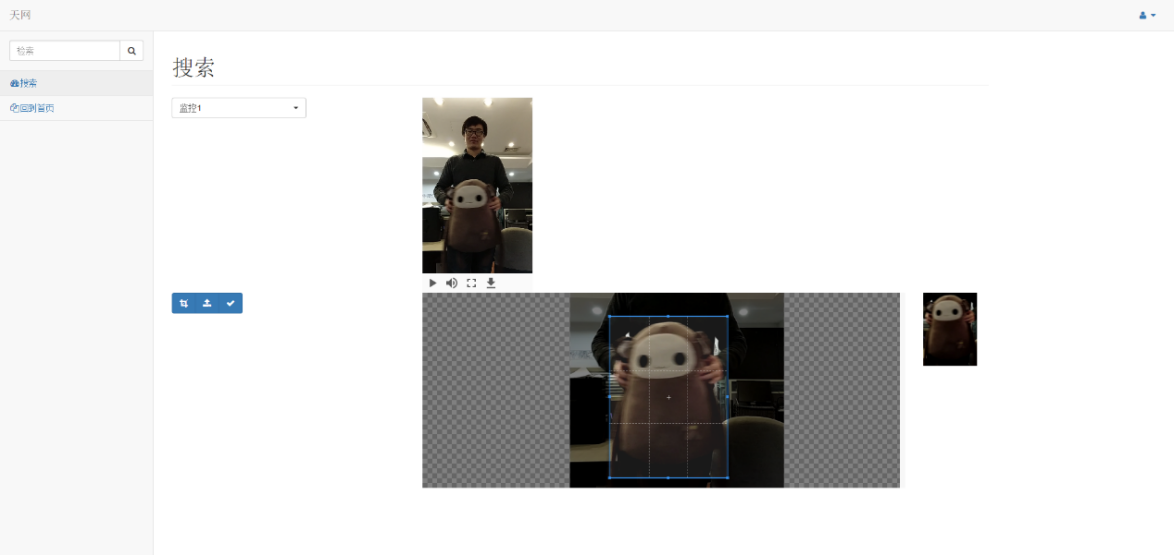


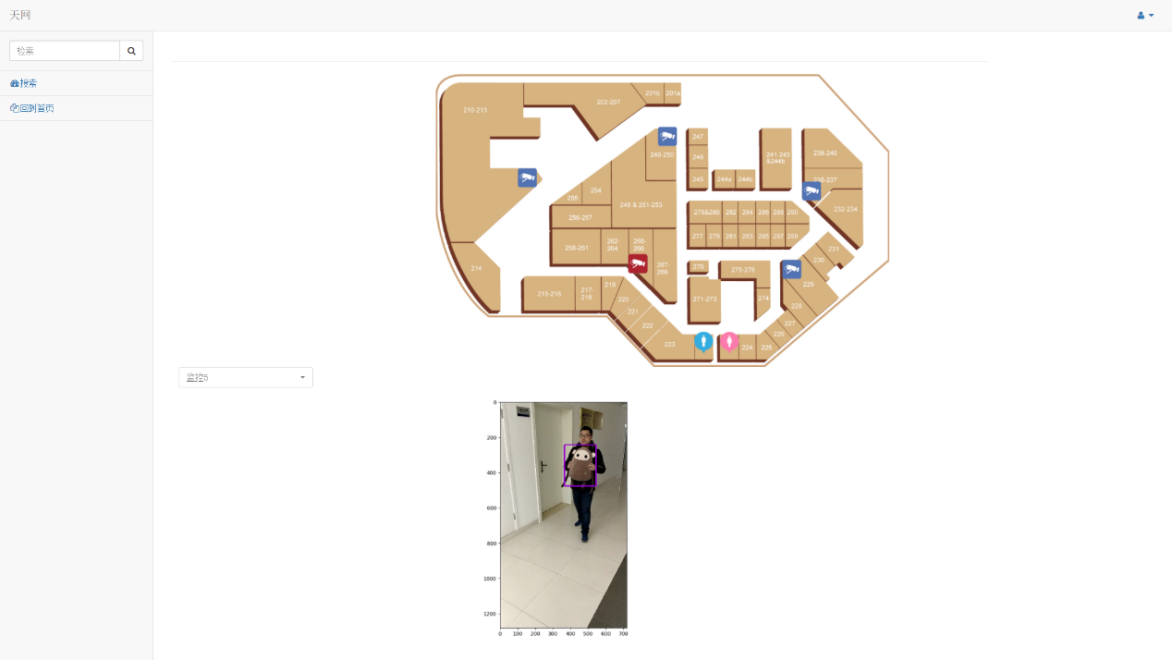
查看实时监控：在打开平面图以后，用户可以通过在地图中点击摄像头，使被选中的摄像头高亮显示，并查看该摄像头的实时视频。



查看历史监控视频：选取某个摄像头，可以播放其历史监控视频。

根据图片搜索目标：用户可以在历史视频中定格并框选得到目标对象（如下图所示，通过在某张截图中框选玩偶作为搜索目标），也可以自行上传图片并框选出目标对象。系统会根据搜索目标出现的位置和行进速度，计算出可能的范围，对范围内的摄像头的视频进行搜索，找出搜索目标的位置，如下图所示：





### 2.1.2“查看摄像头参数”用例规约

描述：用户进入后即可在首页查看全区域平面图与摄像头信息。

执行者：用户

基本流：

1. 系统显示建筑物平面图、摄像头位置及摄像头覆盖位置

2. 用户将鼠标移至某一摄像头上方

3. 系统显示该摄像头相关参数、位置信息并高亮其覆盖位置

备选流：

2a. 用户查看平面图后直接登出

用例结束

3a. 该摄像头数据无法获取

系统提示“该摄像头数据异常”，返回1

非功能需求：系统响应用户需求时间不超过3秒

业务规则：3a. 系统同时最多只能显示一个摄像头的详细信息（相关参数与位置信息）

### 2.1.3 查看实时监控”用例规约

描述：用户通过在前端页面显示的地图中点击摄像头，被选中的摄像头高亮显示，并给出摄像头的实时视频

执行者：用户

基本流：

1. 用户浏览界面并选择摄像头，单击选中的摄像头

2. 显示对应摄像头的实时视频，被选中的摄像头高亮，前端页面进行调整

备选流：

1a. 用户没有选中摄像头

返回1

2a. 对应摄像头数据调取失败

系统提示“该摄像头数据异常”，返回1

2b. 用户选中其他摄像头

返回1

前置条件：系统位于首页

后置条件：若正常反馈则显示摄像头的实时视频

非功能需求：系统响应的时间不超过3秒

业务规则：2a. 同时最多只能显示一个摄像头的画面

### 2.1.4“查看历史监控视频”用例规约

描述：用户可以在选取某个摄像头后播放其历史数据

执行者：用户

基本流：

1. 用户单击选中摄像头

2. 系统高亮选中摄像头

3. 用户点击“查看历史视频”

4. 系统播放该摄像头的历史视频

备选流

2a. 用户点击其它摄像头

系统切换到选中摄像头，回到2

4a. 对应摄像头数据调取失败

系统提示“该摄像头数据异常”，返回1

前置条件：系统在查看平面图界面

后置条件：用户可在观看视频时任意暂停、继续、快进、快退

非功能需求：系统响应的时间不超过3秒

业务规则：3a. 用户仅能查看距当前时间一周以内的历史视频

### 2.1.5“根据图片搜索目标”用例规约

描述：用户可以在摄像头的历史数据或自行上传的图片中选定搜索目标，根据用户输入图片，系统在历史视频以及直播视频流中搜索，并返回可能位置。

执行者：用户

基本流：

1. 用户调用“查看历史数据”用例，观看该摄像头历史视频并选择一帧暂停

2. 用户点击“截取”按钮，在视频上截图

3. 系统展示被截取图像，询问是否确认上传

4. 用户选择确认上传

5. 系统显示选择成功

6. 用户选择单层搜索

7. 根据用户上传图片，返回对象可能位置显示对应摄像头的实时视频，被选中的摄像头高亮，前端页面进行调整

备选流：

1a. 用户点击“本地上传图片”

系统在截取图像界面显示用户选择的图片，进入2

1b. 用户没有选择暂停，视频播放完毕

用例结束

2a. 用户选择放弃

用例结束

6a. 用户选择多层搜索

系统根据图片找到对象在所有楼层的可能位置，返回对应摄像头的实时视频，并高亮被选中摄像头

7a. 系统无法找到任何可能位置

系统提示“找不到目标”，用例结束

前置条件：系统位于首页

后置条件：能返回对象可能位置

业务规则：7a. 可显示任意数量的摄像头视频

## 2.2. 易用性

系统应方便用户使用，有基础计算机水平的用户学习时间应不超过5分钟；同时系统提供在线支持帮助，以方便用户使用该系统。一个学习过的用户应该可以在平均3分钟或最多5分钟时间以内完成人员搜索操作（系统搜索时间不计）。

## 2.3. 可靠性

**2.3.1 可用性：**系统必须保证每天24小时不间断运行，年平均正常运行时间达到99.5%，同时应当正确处理发生的异常或错误，并返回错误信息

**2.3.2 MTBF：**约一个月

**2.3.3 MTTR：**约一小时

**2.3.4 精确度：**系统视频输出视频分辨率720p，识别准确率在70%以上

**2.3.5 最高错误或缺陷率：**不大于60bugs/KLOC

**2.3.6 错误或缺陷率：**

小错误：指页面显示异常，网页的某部分无法显示。错误率<10%。

大错误：指网页对客户操作无响应。错误率<3%.

严重错误：指系统崩溃、数据库崩溃等令整个软件无法工作的错误。错误率应为0。

## 2.4. 性能

**2.4.1 响应时间：**响应时间不超过10s

**2.4.2 吞吐量：**考虑到服务器性能，系统应支持10个并发用户同时搜索，其余搜索请求将加入等待队列

**2.4.3 容量：**系统最多可支持100名用户同时在线

**2.4.4 降级模式：**系统降级运行时，可查看实时监控和历史记录，无法进行搜索

**2.4.5 资源利用情况：**系统应有至少16GB内存与1TB硬盘空间可供使用，同时应与监控系统、数据库等外界系统保持通信，通信速度应不低于1MB/s

## 2.5 可支持性

**2.5.1 编码标准：**系统编码使用utf-8字符集，系统语言使用中文

**2.5.2 命名约定：**变量命名使用驼峰式命名法

# 3.架构设计思想

B/S模式，统一客户端，简化系统开发、维护和使用。

系统整体采用微服务模式。将一体化应用分解、模块化，变成多个单体应用，易于理解、开发与维护；项目组内部不同服务的负责人可以自由选择技术，最终只需提供对外的api接口；每个服务独立部署，利用docker容器技术与kubernetes等容器调度技术，在云端可以轻松实现集群部署。

后端系统采用分层模式。面向接口的编码，以之为根据进行详细设计，实现各层之间的解耦。

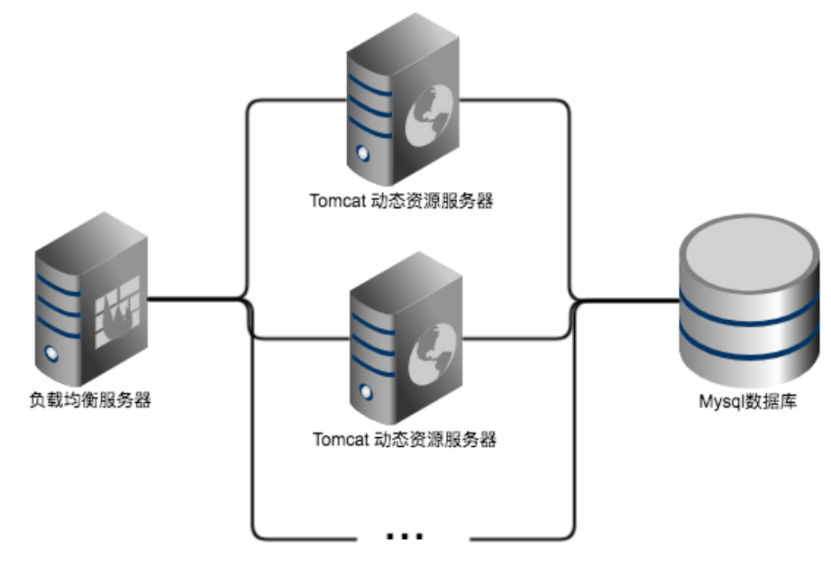
# 4. 架构体系描述

GETS系统主要分为以下几个服务：视频直播转发服务、动态资源服务、识别服务、静态资源服务。每个服务暴露api进行通信。

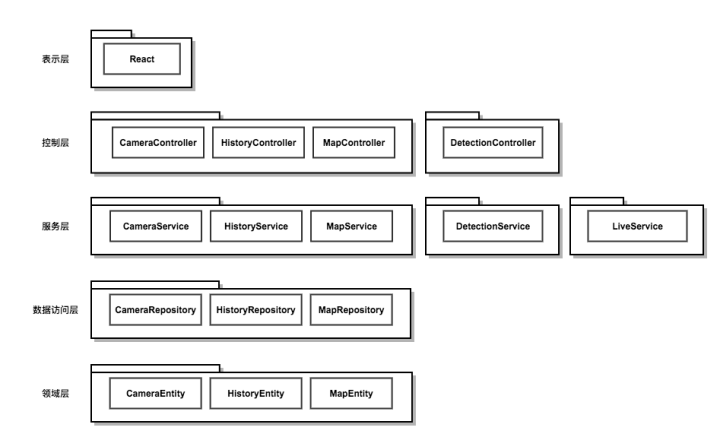
其中后端分层分为：数据访问层、业务层、控制层。通过分层，定义层与层之间的接口，进行层级之间的解耦，增强可扩展性、可维护性，降低集成难度，提高效率。

# 5. 架构图

## 5.1部署视图structure-2.0.png

其中对Nginx静态服务与Tomcat动态服务做负载均衡:

## 5.2逻辑视图



### 5.2.1 概述

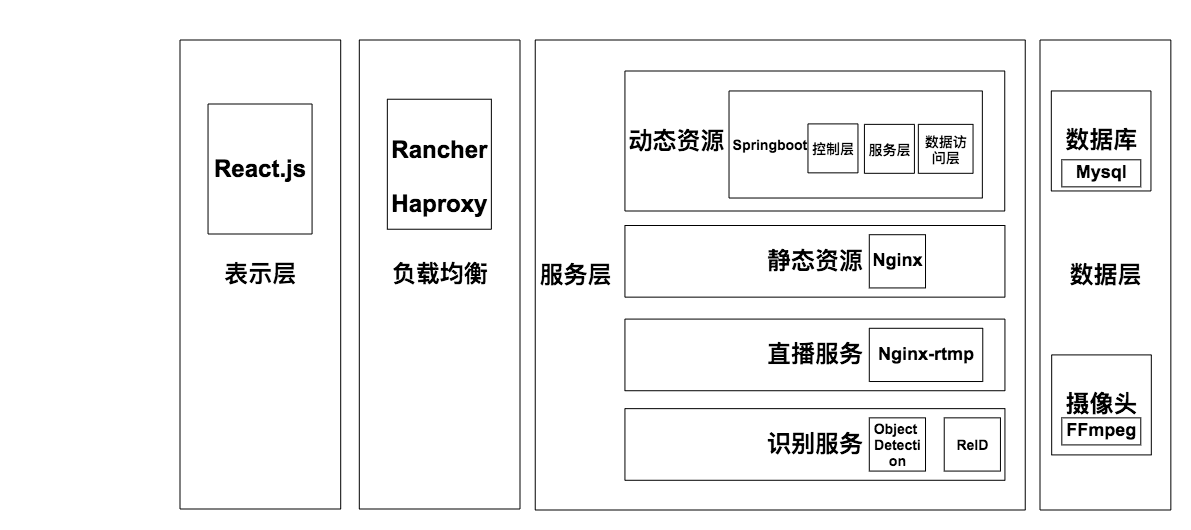
整体分为五层结构。表示层、控制层、服务层、数据访问层。

主要代码在表示层React部分，以及服务层的业务逻辑。

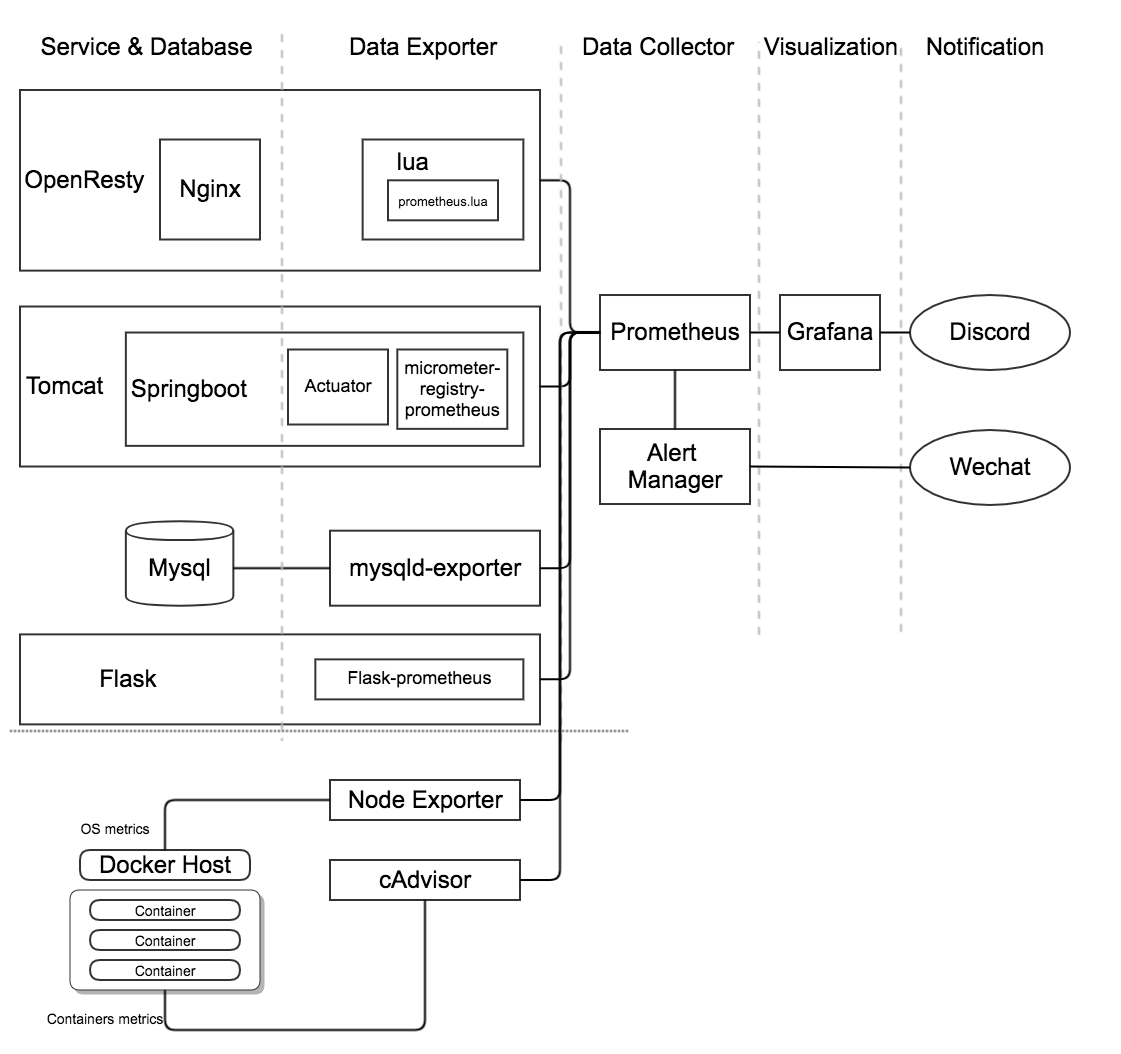
### 5.2.2具有重要意义的设计包

1. 表示层页面
   1. 主要职责：页面渲染，用户通过网页访问服务
2. Detection Service 识别服务
   1. 主要职责：从历史视频/视频直播中识别对象
   2. API：详见接口设计
3. Live Service 直播服务
   1. 主要职责：视频直播转发
   2. API：详见接口设计

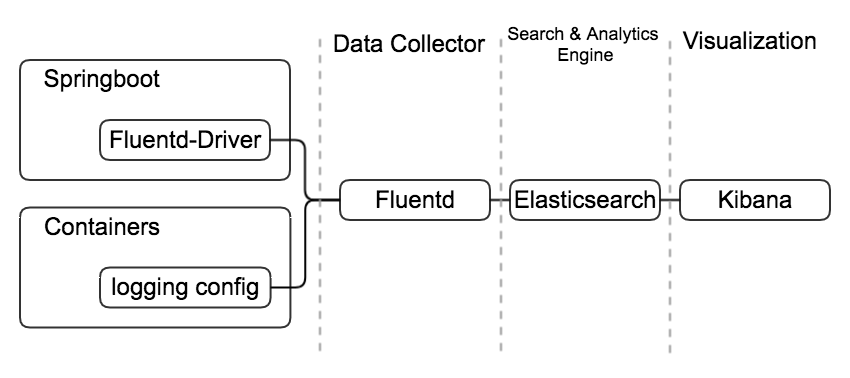
## 5.3 实现视图



## 5.4 运维监控架构



## 5.5 日志管理架构



# 6.技术选型

- 技术选择

- 前端： React.js, OpenResty

- 后端： Java, Springboot, Tomcat

- 识别服务： Python, TensorFlow, Flask

- 直播服务： Nginx-rtmp

- 数据库： Mysql

* 集群管理：Rancher
* CI/CD: Rancher Pipeline

- 版本管理： Github

- 监控系统：Prometheus

- 监控面板：Grafana

- 自动报警：Alert Manager

- 日志收集：Fluentd

- 日志搜索：Elasticsearch

- 日志面板：Kibana

- API管理：Swagger

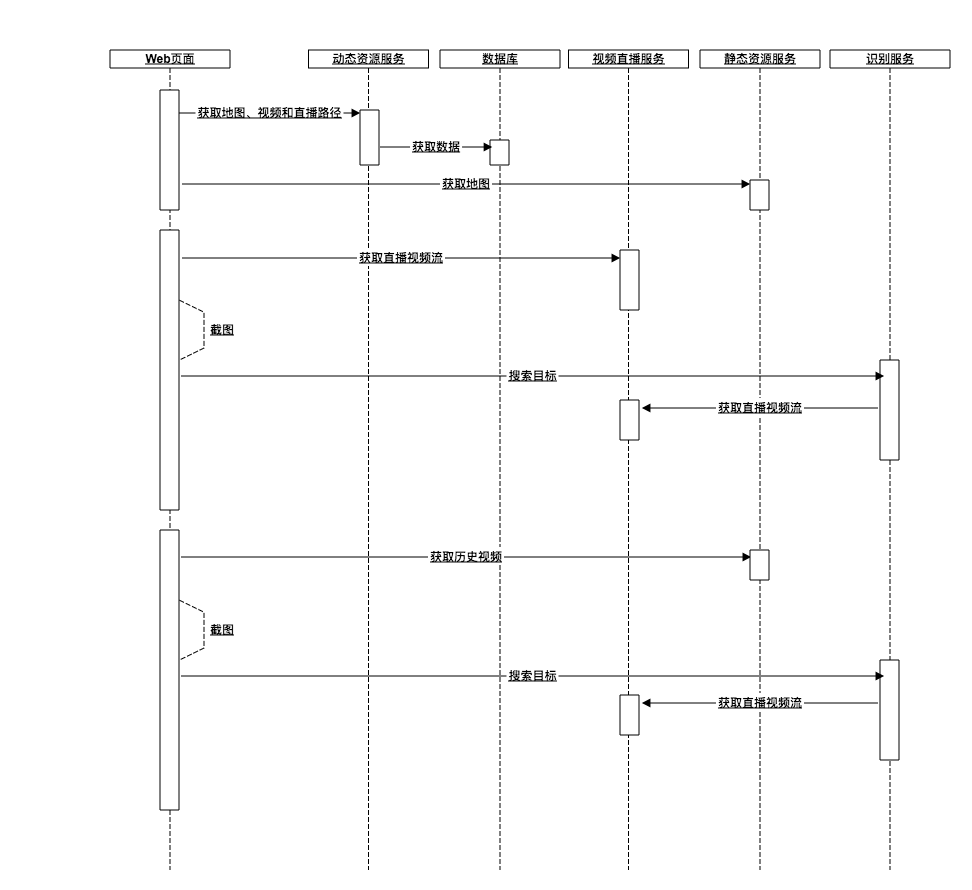
- 运行环境： Chrome

# 7.过程视图

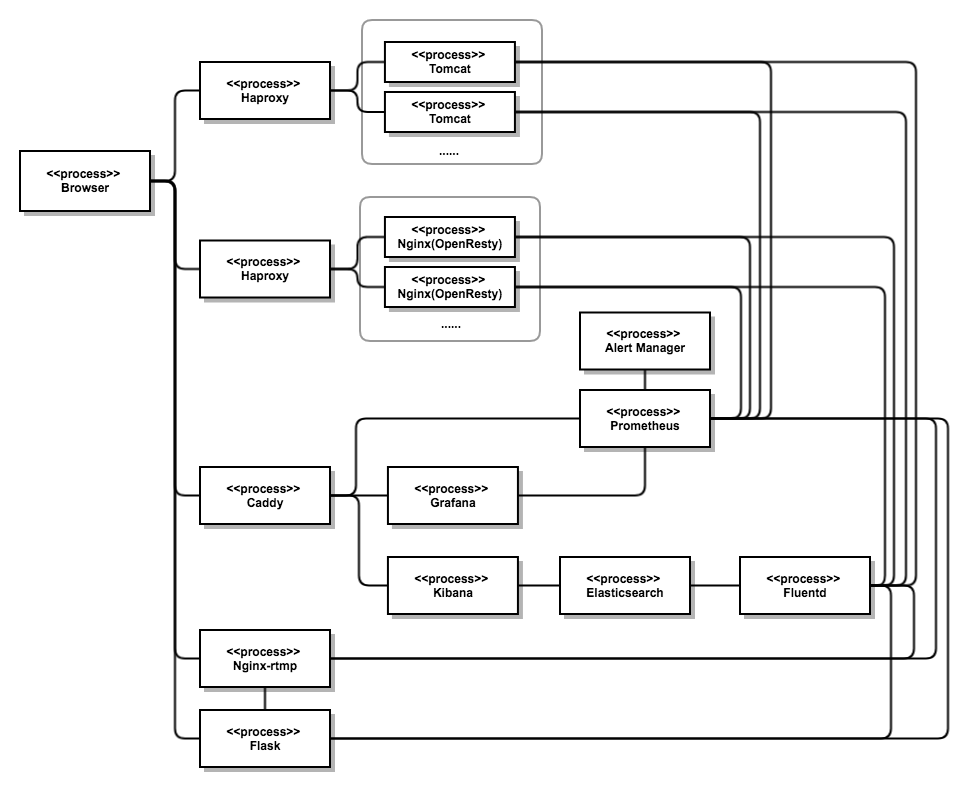
## 7.1 通信协议

大部分服务以HTTP协议暴露 api，视频直播传输使用标准rtmp协议

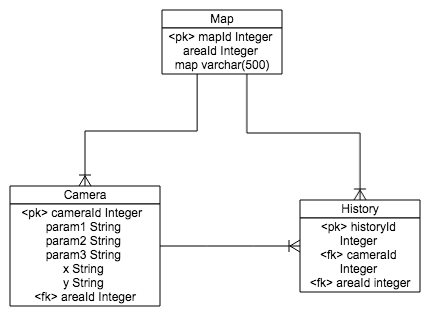
## 7.2 时序图



## 7.3 进程视图



# 8.数据视图

类图

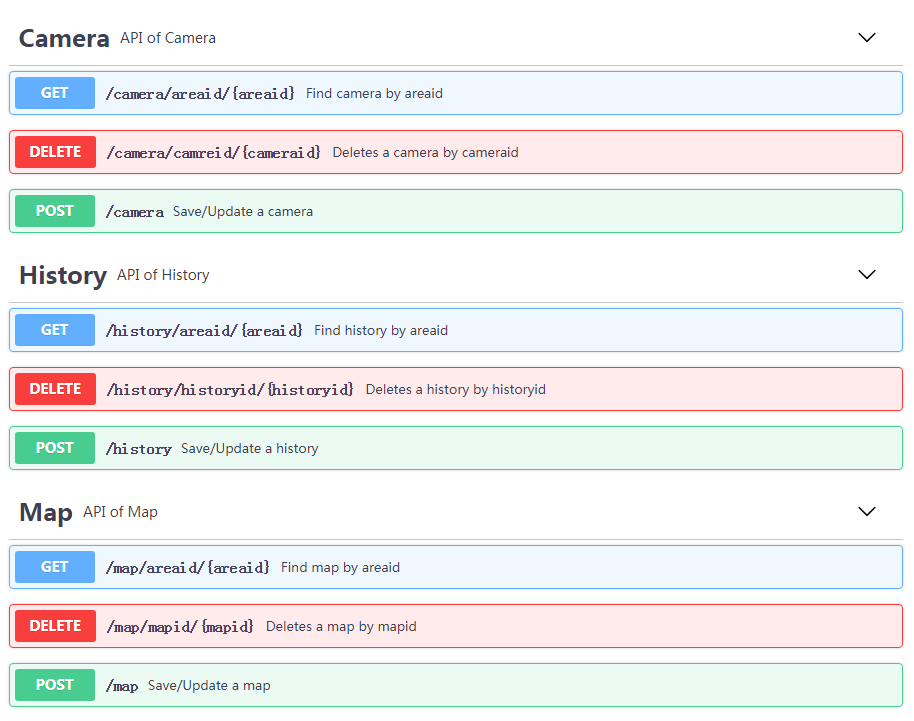
# 9 接口设计

## 9.1 动态资源服务接口

• 协议: HTTP

• port: 8081 (default)

• 详情见 code/swagger/swagger.yaml 文件，在swagger editor 中查看，效果如下图所示。可以查看返回类型，并且发请求测试API。



## 9.2 视频直播服务

### 9.2.1 视频推流

• 协议: RTMP

• url: /stream/{$STREAM\_NAME}

• port: 1935

example: rtmp://127.0.0.1:1935/stream/hello

### 9.2.2 视频播流

• 协议: HLS

• url: live/{$STREAM\_NAME}.m3u8

• port: 8080

example: <http://127.0.0.1:8080/live/hello.m3u8>

虽然HLS本身实现会导致20s左右延迟，但是其他播放协议均需要flash支持，而HLS只需要html5即可实现视频转发。苹果移动设备不支持flash。

## 9.3 静态资源服务

• 协议: HTTP

• port: 80

### 9.3.1 静态网页

• url: /

### 9.3.2 视频文件

• url: /vod/{$VIDEO\_NAME}

## 9.4 识别服务

• 协议: HTTP

* port：5000

### 9.4.1 从直播中进行搜索

• url: /stream

* method: POST
* request body parameters

img – 截图的base64编码

• return: JSON（status；filename；picture；time）

picture - 搜索结果的JPG图片base64编码（无编码头）

### 9.4.2 从历史视频中进行搜索

• url: /history

* method: POST
* request body parameters

img – 截图的base64编码

• return: JSON（status；filename；picture;time）

picture - 搜索结果的JPG图片base64编码（无编码头）

**9.5 监控与可视化**

**9.5.1 Prometheus Data Exporter**

端口不对外开放，使用内网ip访问

- Nginx

- port: 9527

- url: /metrics

- Mysql

- port: 9104

- url: /metrics

- Tomcat

- port: 8081

- url: /actuator/Prometheus

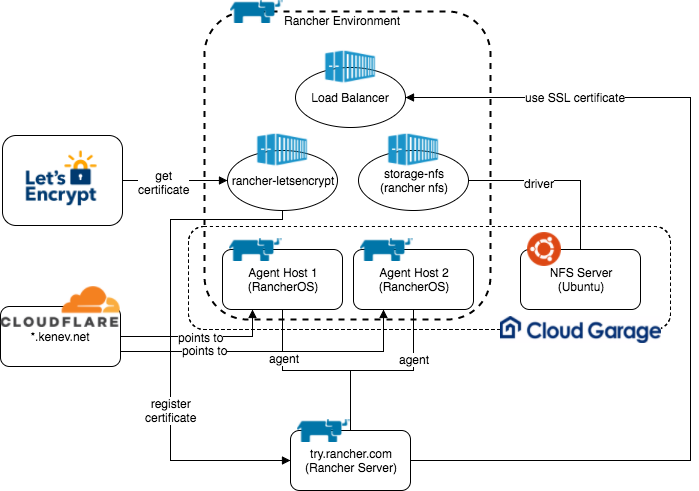
**9.5.2 Prometheus**

port: 9090

**9.5.3 Grafana**

port: 3000

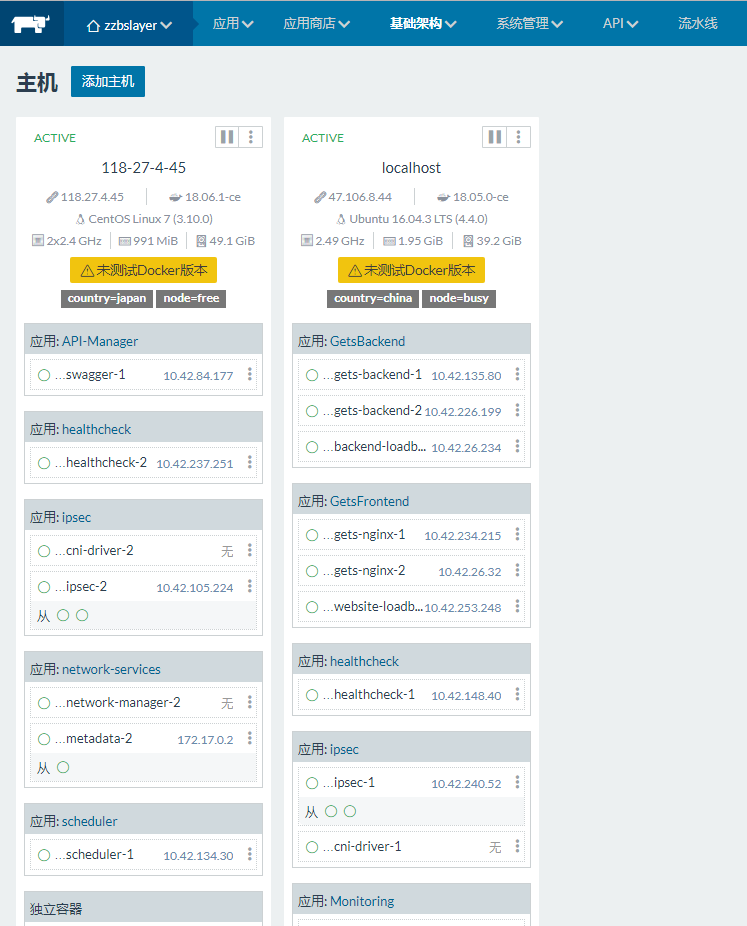
# 10 部署运维



## 10.1集群管理

使用rancher进行容器的集群管理、调度。

### 10.1.1 主机节点



一台国内阿里云主机作为rancher-server，只作为集群管理的master节点；一台国内阿里云主机作为rancher-agent，大多数docker都运行在这台主机上（之后阿里云节点均代指该agent节点）；一台日本conoha主机作为rancher-agent，供rancher调度，防止作为rancher-agent的阿里云主机挂掉。

每个rancher agent自带healthcheck, ipsec, network-services, scheduler等功能。在rancher界面，通过导入docker-compose.yml, rancher-compose.yml 即可生成复数镜像以及负载均衡功能。

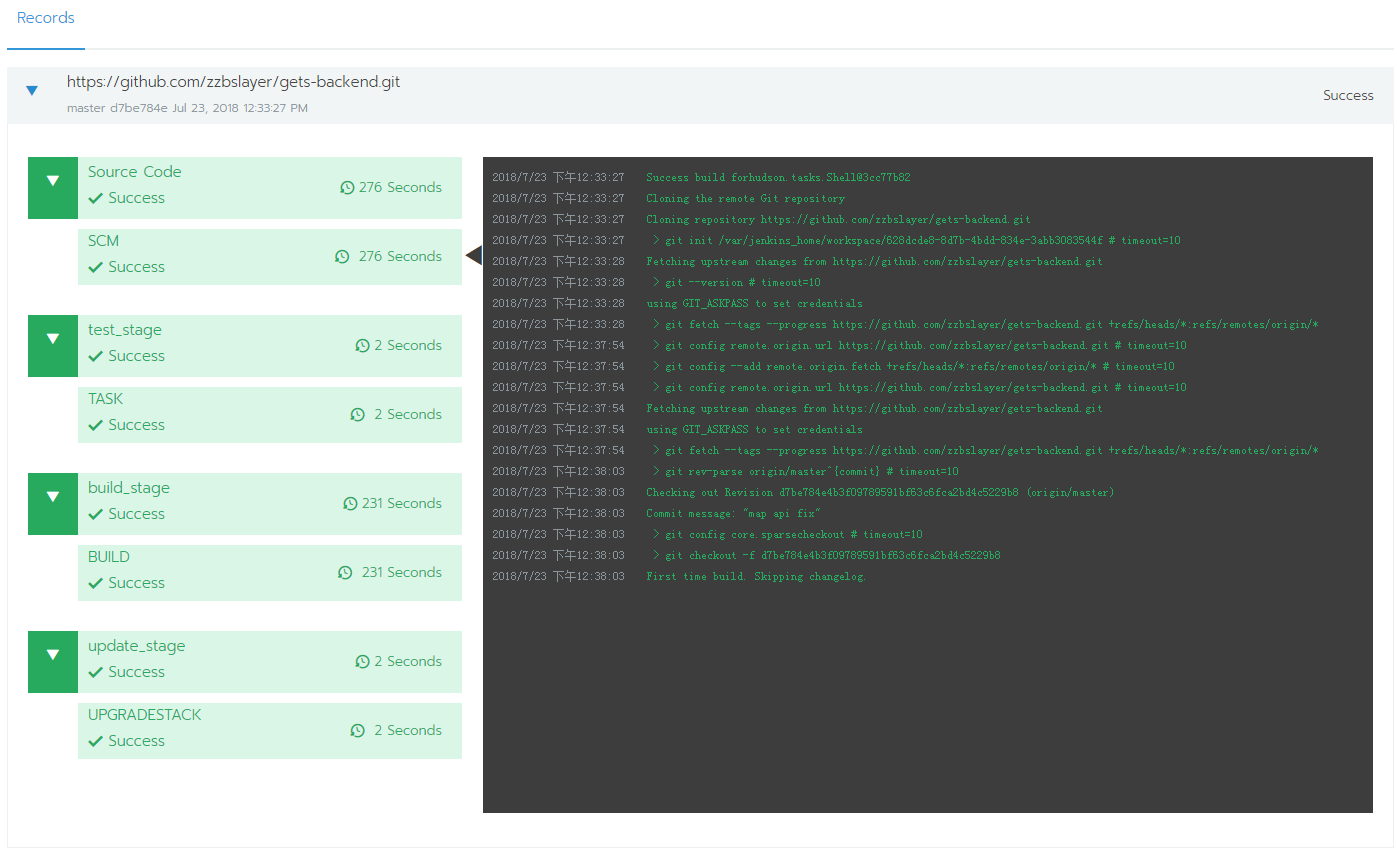
### 10.1.2 镜像分布

阿里云节点：Nginx 静态资源服务器，Tomcat动态资源服务器，Mysql数据库，Nginx视频转发服务器，Prometheus

Conoha节点：Swagger Editor，Grafana

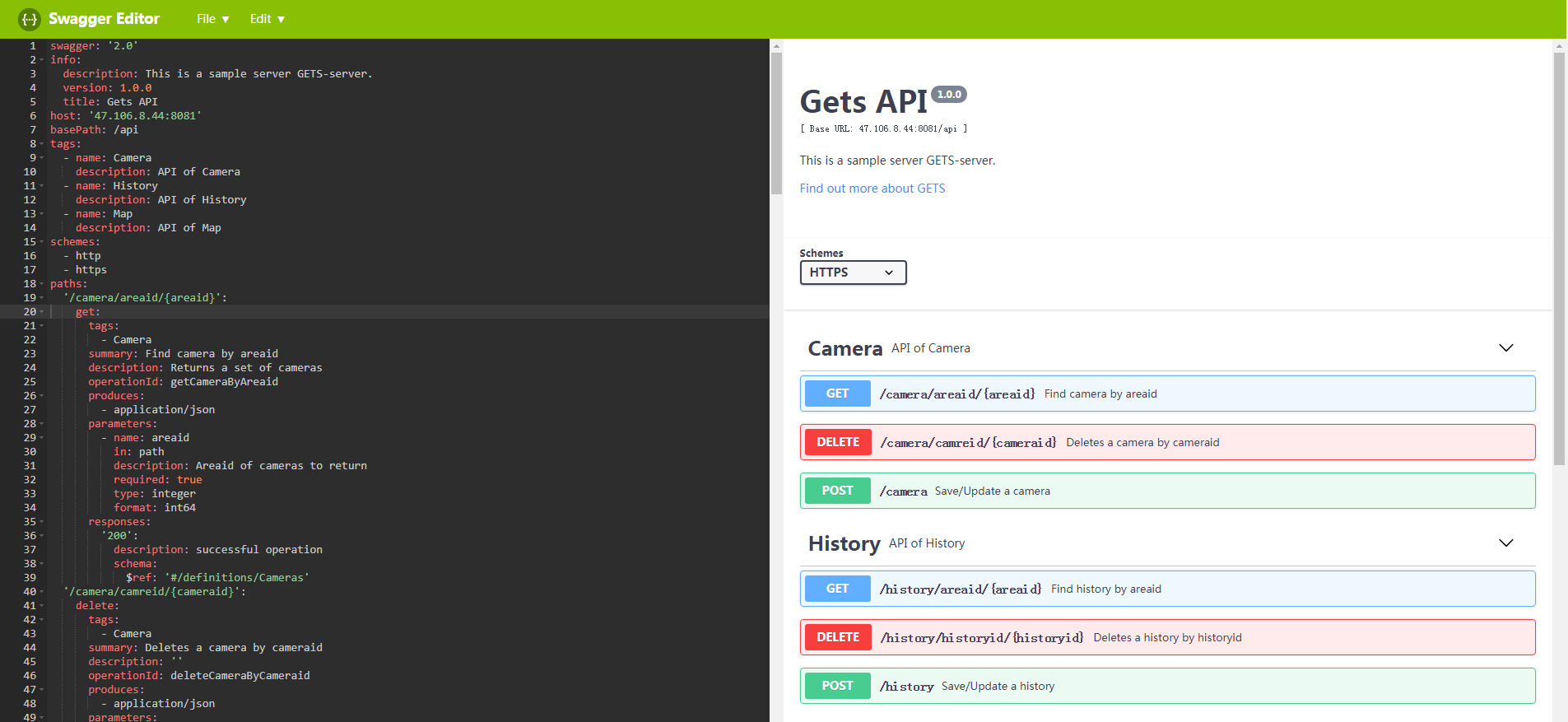
## 10.2 CI/CD

利用rancher pipeline 实现CI/CD 全自动流水线，并且不间断更新运行中的镜像。不过CPU使用率



## 10.3 API 管理

利用swagger管理、查看api。用.yaml文件来描述api的url、请求类型、参数类型、返回类型等。并且可在网站上测试API

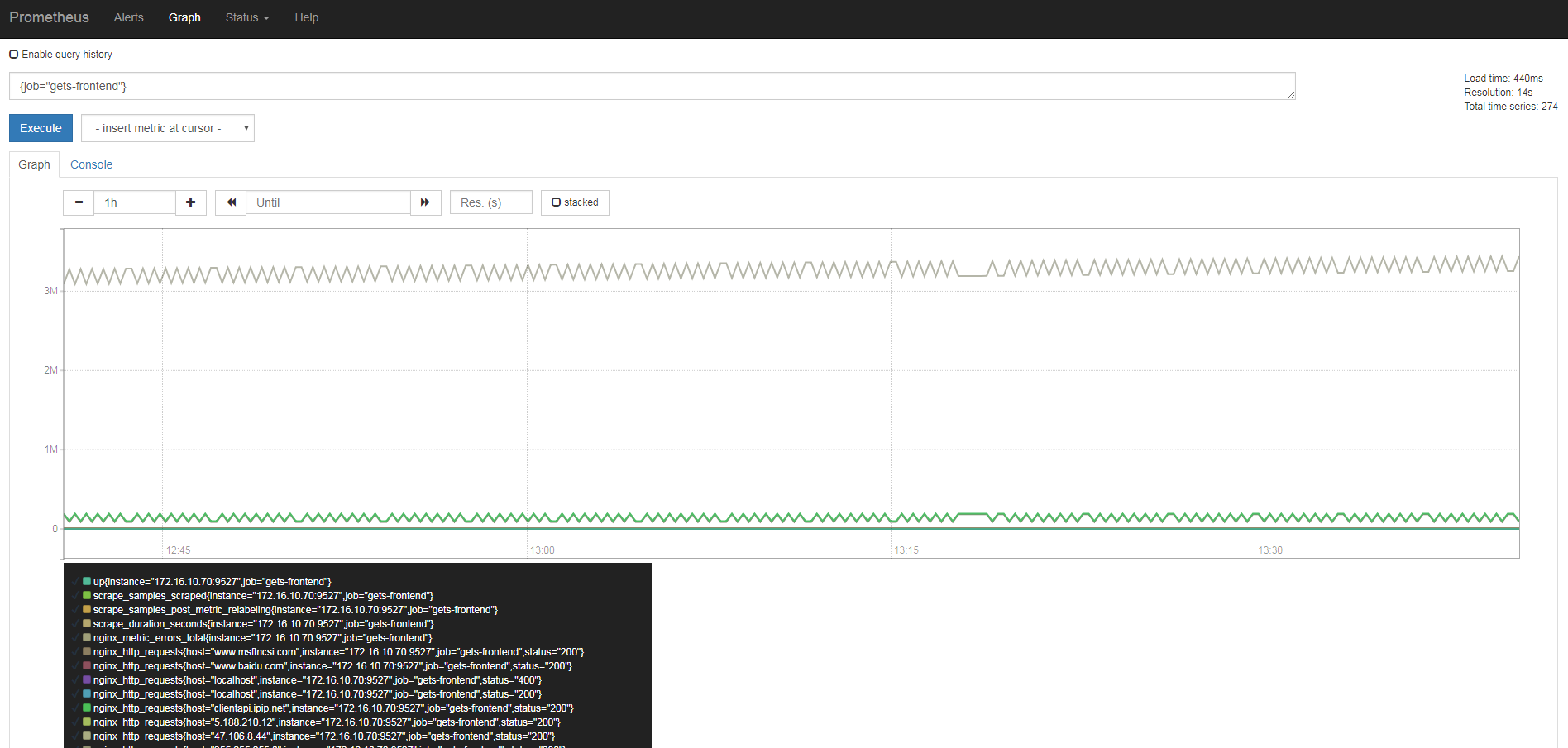


## 10.4 数据监控与通知

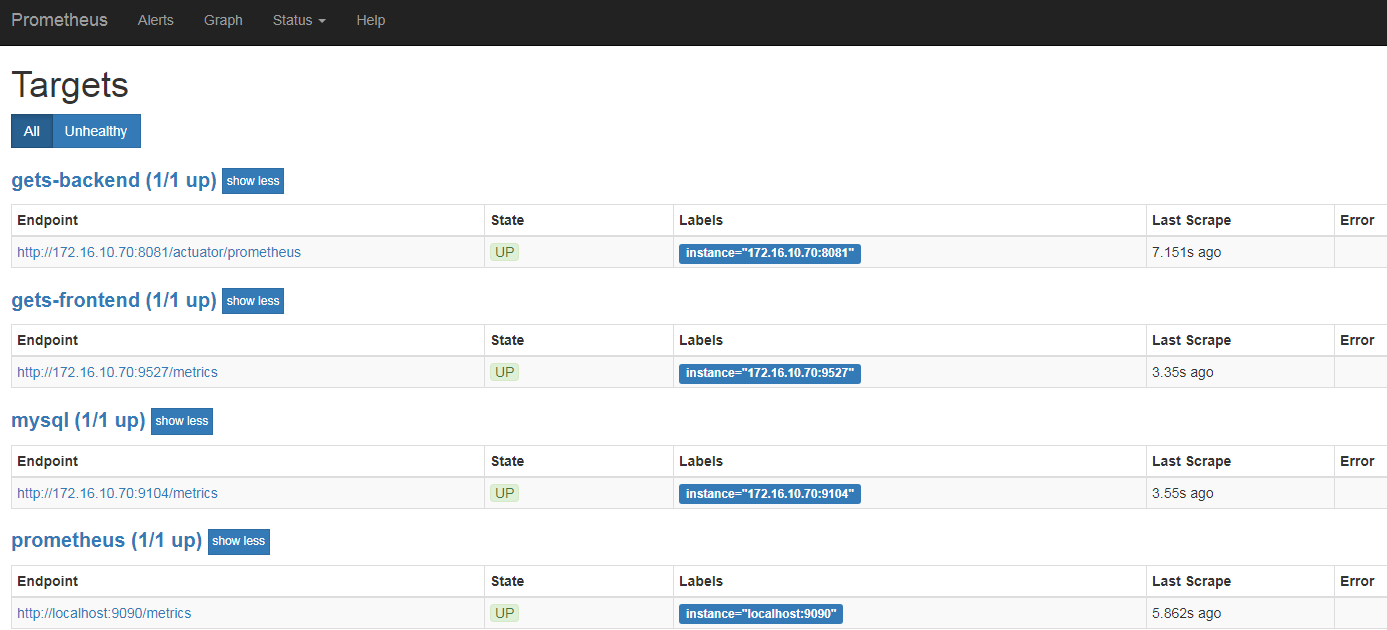
### 10.4.1 Prometheus

Prometheus是一个监控系统以及时序数据库。在本项目中，只作为监控数据收集与存储的作用。（详见运维监控架构）

每个服务配置对应的data exporter，通过http将Prometheus格式的监控数据输出。然后在Prometheus配置文件中配置data exporter的url，Prometheus每隔一段时间将data exporter输出数据收集并存储。并可以设置一定的报警规则，触发时通过alert manager自动报警。可以通过http请求在Prometheus运行时，动态修改配置、报警规则等。下图是对静态资源服务器的监控。配置文件在/code/monitoring-system/prometheus下



### 10.4.2 Prometheus data exporter



Data exporter用来输出被监控对象的流量等各项数据。每个被监控对象需要配置的data exporter 不尽相同，上图为Prometheus监控的所有服务以及对应data exporter的健康状态。

1、识别服务使用flask框架，使用flask-prometheus库，在container的8000端口，/metrics路径下输出监控数据

2、后端服务使用springboot，使用actuator和micrometer在container的8081端口，路径/actuator/prometheus下输出监控数据

3、Mysql数据库只需要启动一个mysqld-exporter的docker image即可，在container的9104端口 /metrics路径下输出监控数据

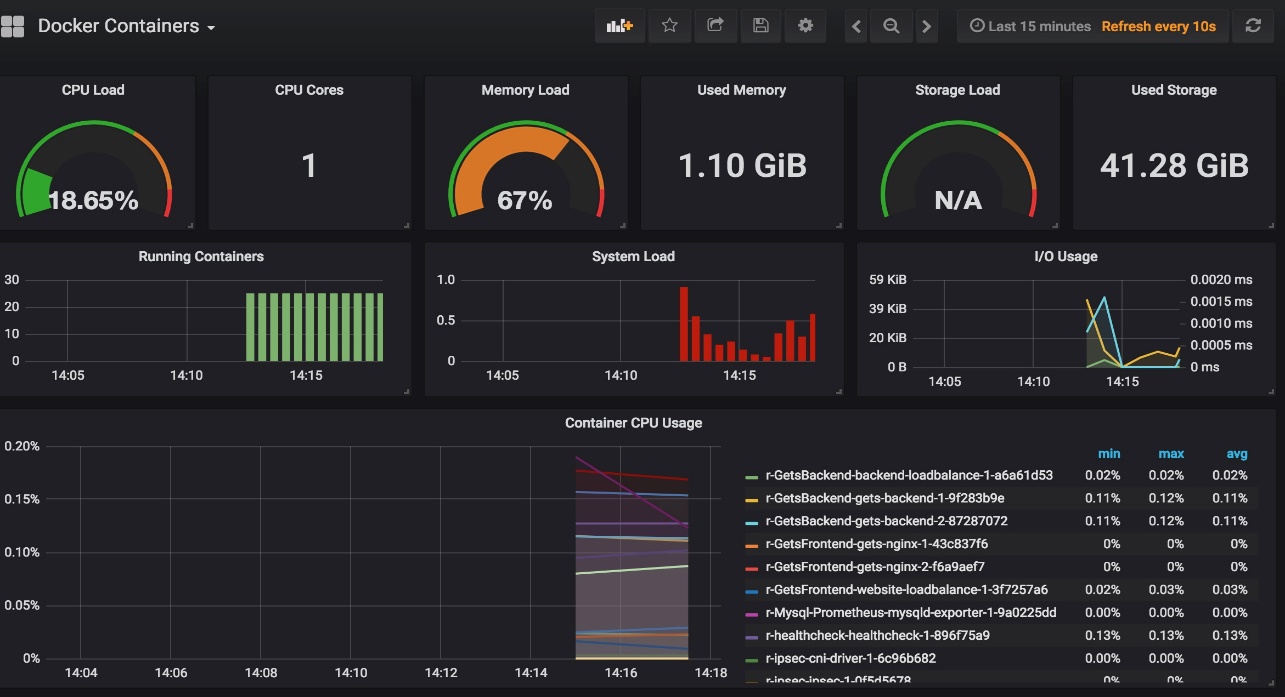
4、静态资源服务器配置最为复杂。Nginx服务器本身不能很好地监控并导出Prometheus格式数据。因此我们抛弃了原始的Nginx，使用OpenResty，一个基于Nginx与lua的web应用平台，安装prometheus.lua插件后，并配置vhost，在container的9527端口 /metrics路径下输出监控数据

5、Node Exporter监控宿主机的情况，在比如CPU使用率、内存占用情况等等，在监控系统的docker network 9100端口，路径/metrics输出监控数据

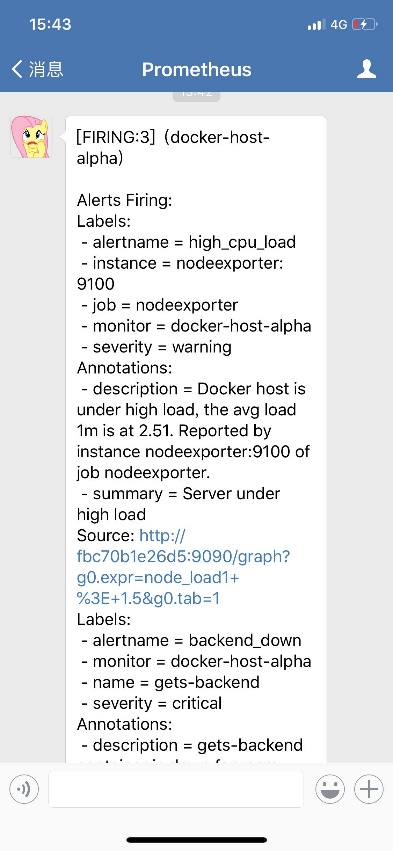
6、cAdvisor监控宿主机中container情况，比如container 内存占用率等等，在监控系统的docker network 8080端口，路径/metrics输出监控数据

### 10.4.3 Grafana

虽然Prometheus自带可视化，但是Grafana的可视化更加优秀。Grafana不仅支持Prometheus的数据源，还支持其他各种来源的数据的可视化，一起集中在Grafana Dashboard里。而且Grafana支持对于Dashboard进行管理、以及JSON配置文件导入导出。下图为Grafana Dashboard。



Grafana也提供报警机制，当触发某些警告规则时会自动提醒。我们的项目配置了Discord的自动报警，与Prometheus报警相比，Grafana报警还能发图片。下图为discord自动报警的测试。配置文件在/code/monitoring-system/grafana下



### 10.4.4 Alert Manager

它能够从Prometheus获取数据，在主界面上显示Prometheus的报警内容，并可以配置通过slack、email、企业微信等方式报警。我们的项目配置了当前端、后端挂掉、cpu占用率过高、内存占用率过高等条件时触发通过企业微信自动报警。右上图为企业微信自动报警。配置文件为/code/monitoring-system/alertmanager/config.yml

## 10.5 日志管理系统

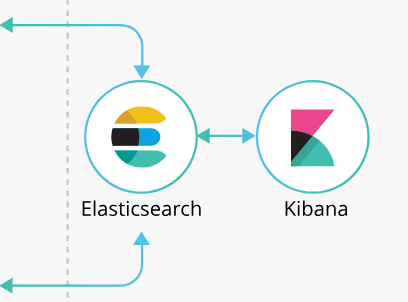
### 10.5.1 Fluentd

Fluentd是一个开源日志收集工具，我们运行一个Fluentd的docker镜像后，向外暴露24224端口，等待日志被采集的对象主动向暴露的端口发送日志信息，然后由fluentd收集。

我们在后端项目中配置fluentd driver，log配置文件中写好将debug级别以上log均输出到fluentd中；在container的docker-compose中配置将log输出到fluentd

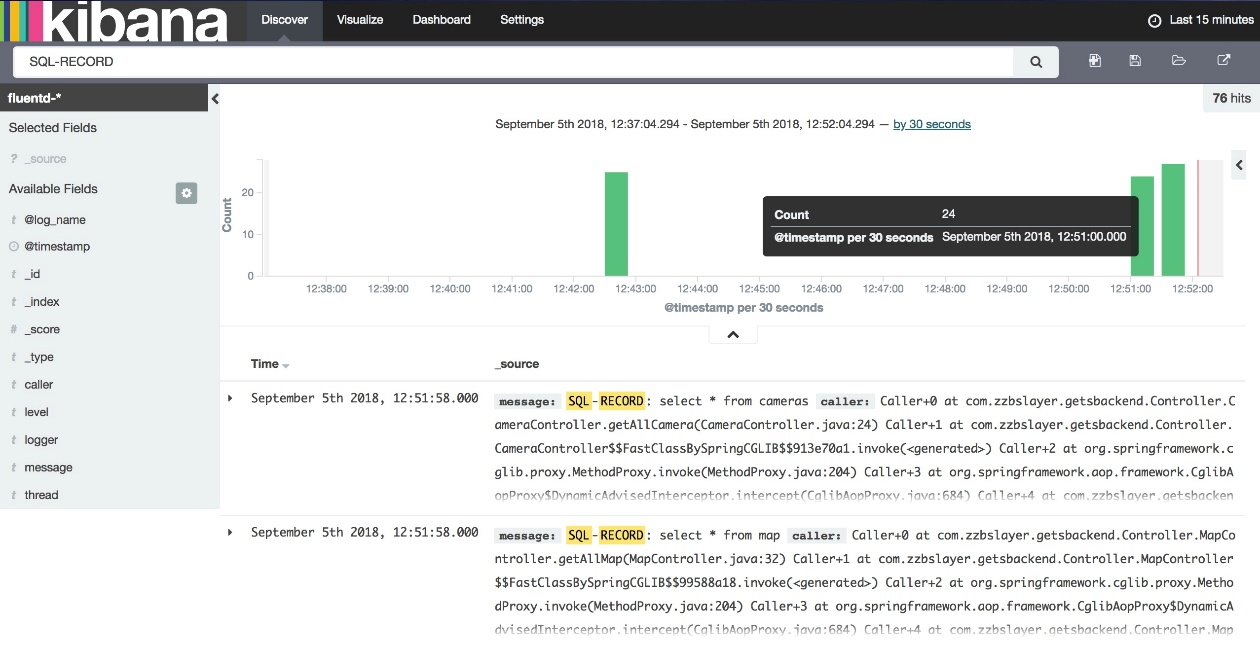


### 10.5.1 Elasticsearch

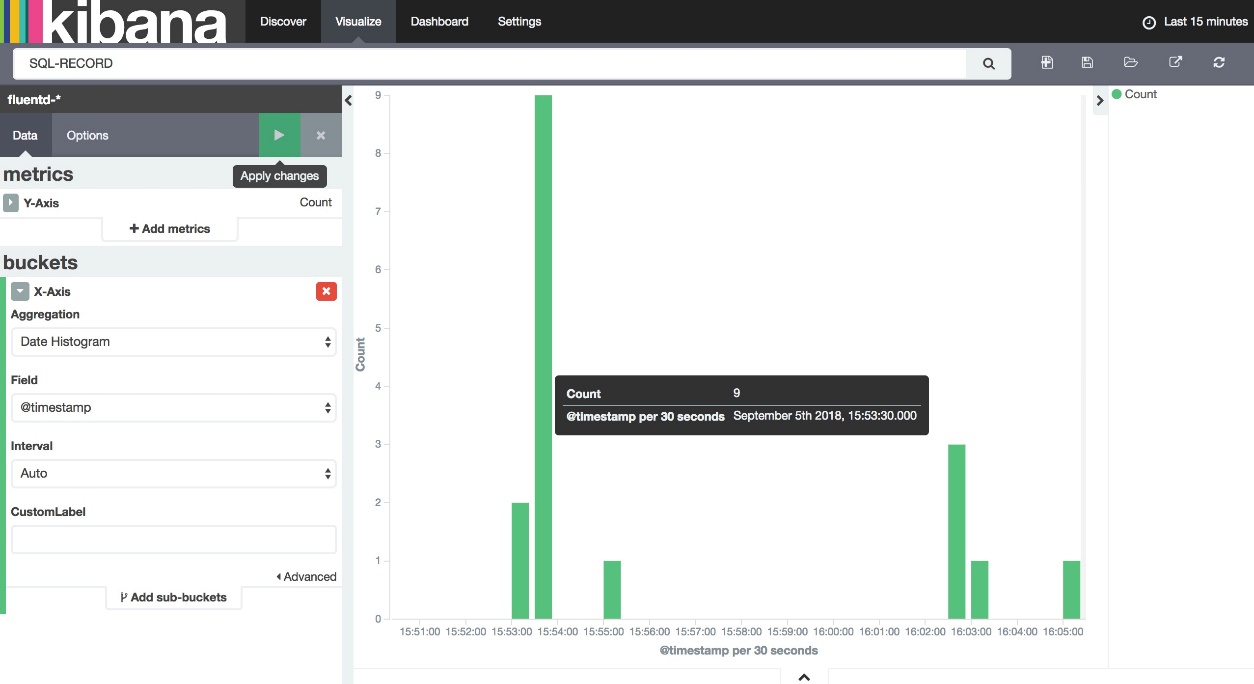
Elasticsearch 是一个分布式的 RESTful 风格的搜索和数据分析引擎。在本项目中从fluentd数据源获得日志数据并存储，暴露9200端口给kibana提供数据源

### 10.5.2 Kibana

首先Kibana为Elasticasearch提供可视化的日志搜索平台。下图为搜索所有带有“SQL-RECORD”关键词的日志。并且可以根据左边各种信息比如log level进行筛选日志。



Elasticsearch使得我们能够搜索、分析日志数据，但是一张好的图表胜过千万行日志。下图为一段时间内SQL查询次数的可视化。



# 11 遗留问题

视频直播 – 由于目前我们使用本地摄像头作为监控，监控录像是从本地向服务器上传，然后网页再向服务器获取监控，会造成不小的延迟。实际应该使用本地服务器以及内网传输，而不是使用云服务器通过公网传输。

性能需求 - 实际并没有高性能服务器提供。个人服务器只有2GB内存，同时2个摄像头进行直播，并且数人同时访问网页观看直播时，其他服务也都在该服务器上跑，服务器有概率卡死。因此性能需求极大程度上受到服务器性能限制。

kubernates集群 - 由于没有那么多节点，而且学生机服务器性能堪忧无法作为k8s master节点，因此无法采用kubernates集群，只能使用rancher默认的cattle

CI/CD – 受服务器网络情况影响，git fetch时常失败。而且十分占用服务器资源，导致服务器宕机，因此之后CI/CD一直处于关闭状态

微服务框架 – 本来想使用微服务框架以及相关工具如istio、consul等，使微服务架构更加完善，同样由于服务器性能原因，时常中途宕机，因此放弃。

# 12 项目成果

Rancher界面 : <http://47.106.36.202:8080>

前端页面：<http://47.106.8.44>

后端api : http://47.106.8.44:8081/api

视频上传api : rtmp://47.106.8.44:1935/stream/{ $INSTANCE\_NAME }

视频直播api : http://47.106.8.44:8080/live/{ $INSTANCE\_NAME + ”.m3u8”}

Prometheus : <http://47.106.8.44:12451>

Grafana : <http://47.106.8.44>:12450

Alert Manager : <http://47.106.8.44:12452>

Pushgateway : <http://47.106.8.44:12453> (部署了pushgateway但实际未使用)

Swagger Editor : <http://118.27.4.45>

Kibana : http://47.106.8.44:5601

Prometheus、Alert Manager、Pushgateway设置了Caddy反向代理，这些服务与Grafana默认账号密码均为admin

Rancher账号：SE\_SUMMER 密码：SE\_SUMMER+!S