

# Prometheus监控研究

平安产险科技中心  
技术架构组  
2017年12月

# 目录

## Content

- >  一、背景概述
- >  二、传统VM的系统监控
- >  三、基于K8S的容器监控
- >  四、容器内外应用监控

### 1.1、研究背景：

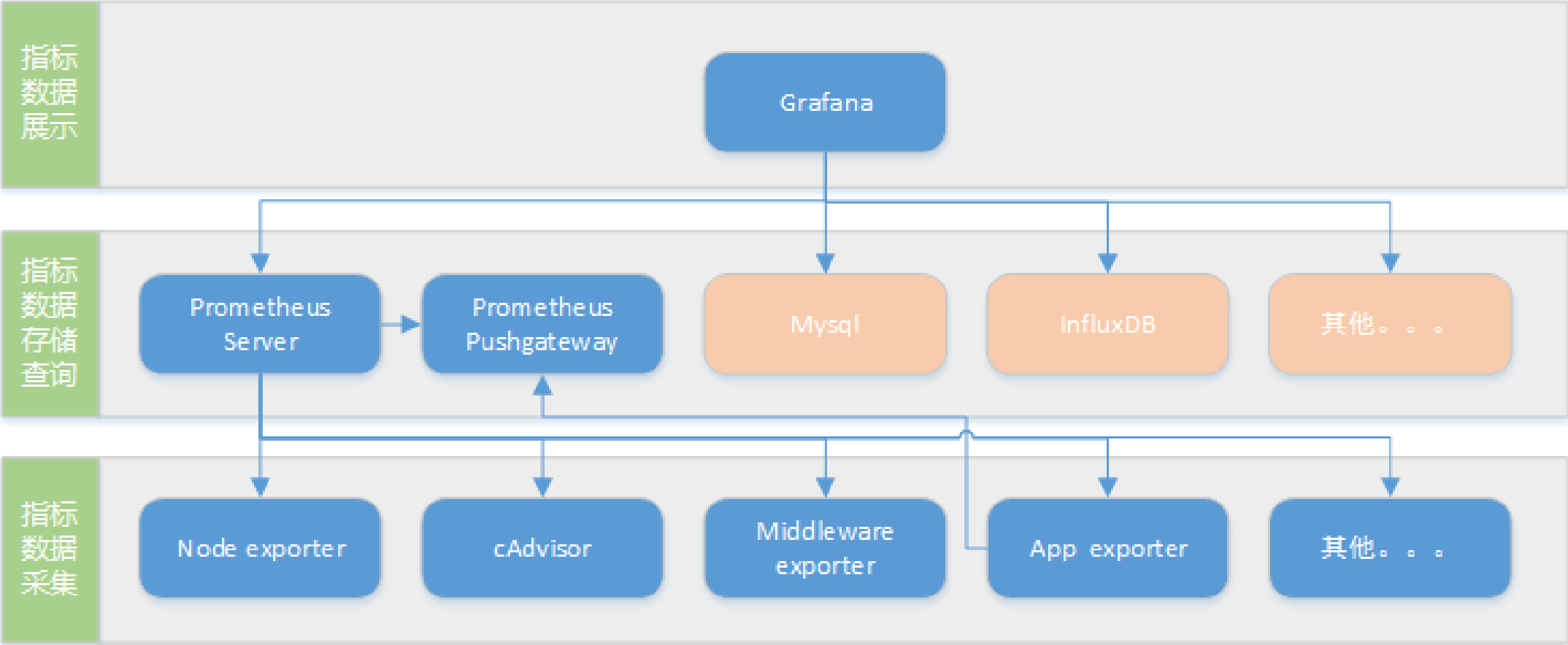
- 现有的监控分散在平安云、中间件、数据库等；需要统一监控入口
- 需要一套能同时兼容传统应用与容器应用的监控系统
- 除了基础资源指标以及中间件指标外，还需要能够进行业务指标监控

### 1.2、研究目标：

- 实现从统一的入口完成IAAS基础设施到中间件、业务指标的全面监控
- 实现同时满足传统虚拟机应用环境监控以及容器应用环境监控
- 实现除了基础的计算、内存、网络等基础指标以及中间件、数据库指标外还能够对应用的业务指标实现灵活监控

	Docker ps/top/stats	Sysdig	Weave Scope	cAdvisor	Prometheus
部署容易度	*****	*****	****	*****	***
数据详细度	***	*****	*****	***	*****
多Host监控	N/A	N/A	*****	N/A	*****
告警功能	N/A	N/A	N/A	N/A	****
非容器监控	N/A	***	***	**	*****

- **选型标准：容器支持、虚拟机传统应用支持、指标采集模块化、展示模块化、容易扩展**
- Docker ps/top/stats 最适合快速了解容器运行状态，从而判断是否需要进一步分析和排查。
- Sysdig 提供了丰富的分析和挖掘功能，是 Troubleshooting 的神器
- cAdvisor 一般不会单独使用，通常作为其他监控工具的数据收集器，比如 Prometheus
- Weave Scope 流畅简洁的操控界面是其最大亮点，而且支持直接在 Web 界面上执行命令
- Prometheus 的数据模型和架构决定了它几乎具有无限的可能性。Prometheus 和 Weave Scope 都是优秀的容器监控方案。除此之外，Prometheus 还可以监控其他应用和系统，更为综合和全面



- Grafana支持从多种数据源获取指标数据并进行图表展示
- Prometheus支持各种类型的数据指标采集，包括从虚拟机节点、mysql、redis、nginx等中间件以及应用的定制业务指标数据采集

## 二、传统VM的系统监控 2.2 监控环境搭建及配置

安装参考：

<http://blog.51cto.com/youerning/2050543?from=timeline>

### 1、Prometheus安装

- 解压安装包：tar -zxvf prometheus-2.0.0.linux-amd64.tar.gz
- 启动并加载配置文件：./prometheus --config.file=prometheus.yml

### 2、node-exporter安装

- 解压安装包：tar -zxvf node\_exporter-0.15.2.linux-amd64.tar.gz
- 启动node-exporter：./node\_exporter

### 3、Grafana安装

- 直接rpm安装：rpm -ivh grafana-4.6.3-1.x86\_64.rpm

### 4、其他事项

- 这里只需要安装并启动应用即可，不需要将应用配置为服务

```
scrape_configs:
- job_name: 'prometheus'
  static_configs:
    - targets: ['localhost:9090']
- job_name: 'node'
  static_configs:
    - targets: ['localhost:9100','10.20.26.140:9100']
    labels:
      group: 'monitor'
- job_name: 'mysql'
  static_configs:
    - targets: ['10.20.26.140:9104']
    labels:
      instance: 'mysql140'
- job_name: 'push-metrics'
  static_configs:
    - targets: ['10.20.26.139:9091']
- job_name: 'java_metrics'
  metrics_path: '/prometheus'
  static_configs:
    - targets: ['10.20.26.139:8888']
alerting:
  alertmanagers:
    - scheme: http
      static_configs:
        - targets:
          - "localhost:9093"
```

## 二、传统VM的系统监控 2.3基础虚拟机节点参数监控

### 1、登陆Grafana

- Admin/admin登陆：http://10.20.26.139:3000

### 2、配置Grafana数据源

- 进入：数据源->新增数据源
- 输入数据源名称并选择类型为Prometheus
- 输入Prometheus地址：<http://10.20.26.139:9090>

Name	Prometheus	Default	<input checked="" type="checkbox"/>
Type	Prometheus		

HTTP settings

URL	http://10.20.26.139:9090
Access	proxy



node-exporter

### 3、导入Node exporter模板

- Dashboards->import 导入模板：
- node-exporter-full\_rev8.json

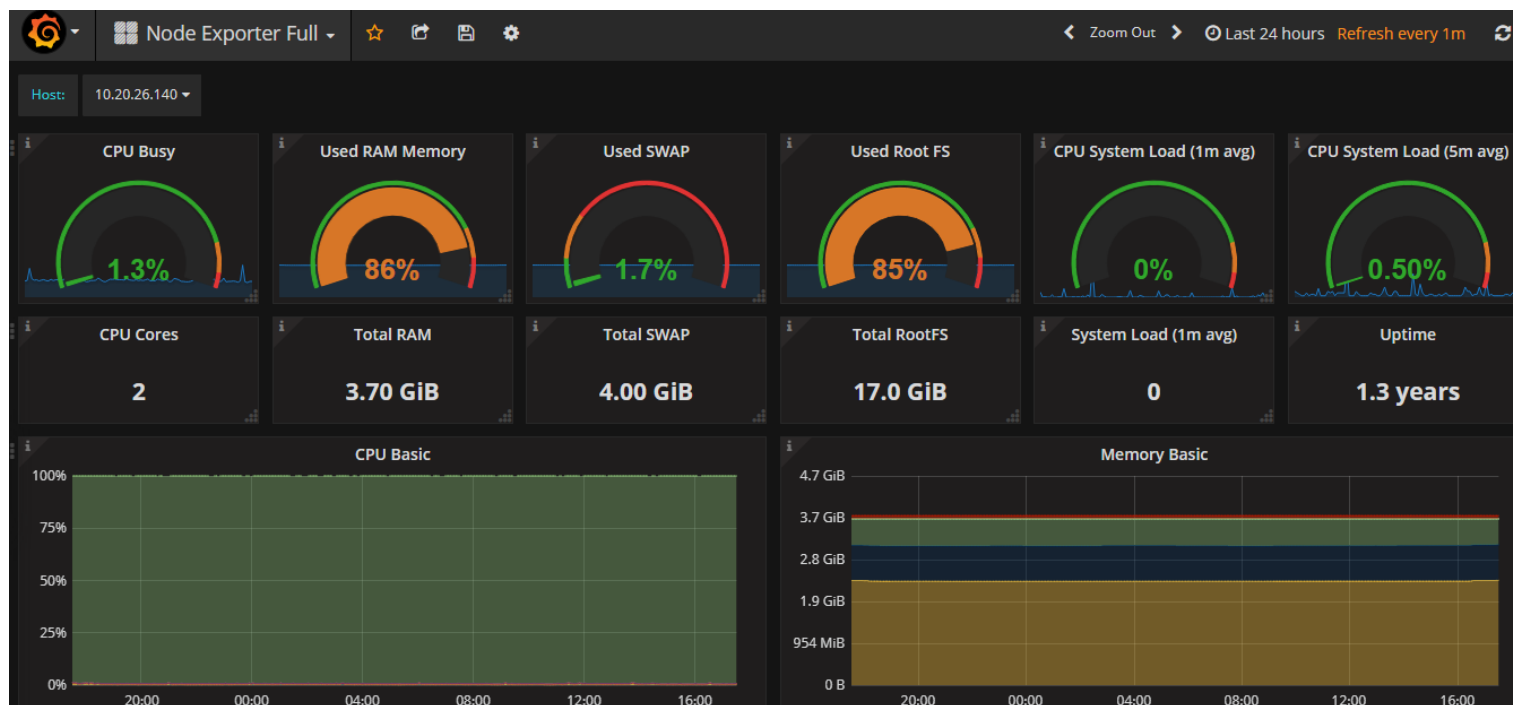
### 4、有用链接：

- 监控目标展示：

<http://10.20.26.139:9090/targets>

- Prometheus监控指标：

<http://10.20.26.139:9090/graph>



# 二、传统VM的系统监控 2.4 中间件系统监控—Mysql为例

## 1、安装exporter：

- 下载：<https://github.com/prometheus>
- 解压mysql exporter
- 解压node exporter

## 2、配置mysql：

- GRANT REPLICATION CLIENT, PROCESS ON \*.\* TO 'prom'@'localhost' identified by xxxxx';
- GRANT SELECT ON performance\_schema.\* TO 'prom'@'localhost';

## 3、编辑配置文件.my.ini：

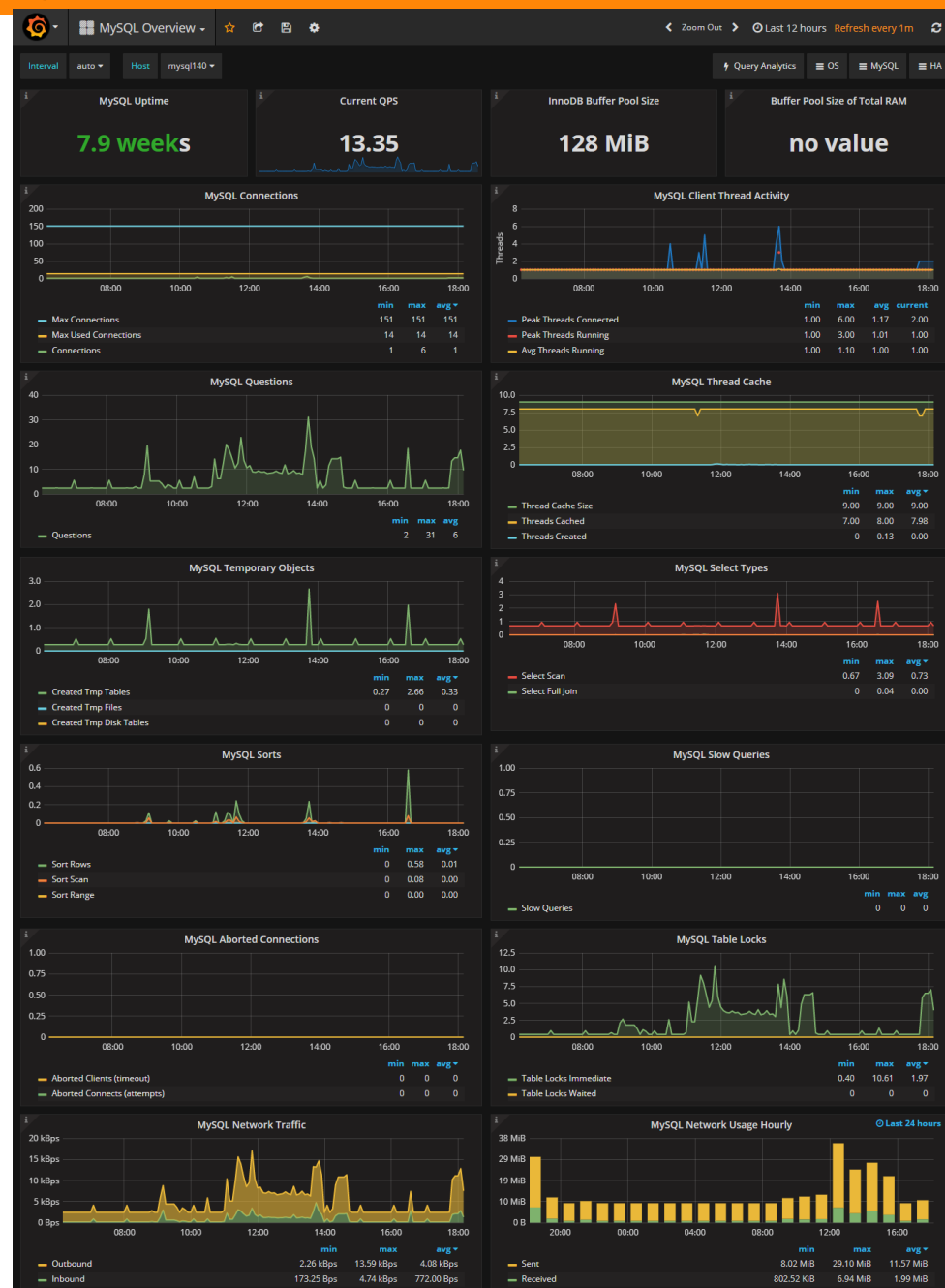
```
[client]
user=prom
password=abc123
```

## 4、启动：./mysqld\_exporter -config.my-cnf=.my.cnf



## 5、Grafana导入Mysql的Dashboard模板

MySQL\_Overview  
w.json





## 二、传统VM的系统监控 2.5 Prometheus支持各种中间件及基础设施的监控

### Databases

- [Aerospike exporter](#)
- [ClickHouse exporter](#)
- [Consul exporter](#) (official)
- [CouchDB exporter](#)
- [ElasticSearch exporter](#)
- [Memcached exporter](#) (official)
- [MongoDB exporter](#)
- [MSSQL server exporter](#)
- [MySQL server exporter](#) (official)
- [OpenTSDB Exporter](#)
- [PgBouncer exporter](#)
- [PostgreSQL exporter](#)
- [ProxySQL exporter](#)
- [Redis exporter](#)
- [RethinkDB exporter](#)
- [SQL exporter](#)
- [Tarantool metric library](#)

### Hardware related

- [apcupsd exporter](#)
- [IoT Edison exporter](#)
- [IPMI exporter](#)
- [knxd exporter](#)
- [Node/system metrics exporter](#) (official)
- [Ubiquiti UniFi exporter](#)

### Messaging systems

- [Beanstalkd exporter](#)
- [Kafka exporter](#)
- [NATS exporter](#)
- [NSQ exporter](#)
- [Mirth Connect exporter](#)
- [MQTT blackbox exporter](#)
- [RabbitMQ exporter](#)
- [RabbitMQ Management Plugin exporter](#)

### Storage

- [Ceph exporter](#)
- [Gluster exporter](#)
- [Hadoop HDFS FSImage exporter](#)
- [Lustre exporter](#)
- [ScaleIO exporter](#)

### HTTP

- [Apache exporter](#)
- [HAProxy exporter](#) (official)
- [Nginx metric library](#)
- [Nginx VTS exporter](#)
- [Passenger exporter](#)
- [Tinyproxy exporter](#)
- [Varnish exporter](#)
- [WebDriver exporter](#)

### Logging

- [Fluentd exporter](#)
- [Google's mtail log data extractor](#)
- [Grok exporter](#)

### APIs

- [AWS ECS exporter](#)
- [AWS Health exporter](#)
- [AWS SQS exporter](#)
- [Cloudflare exporter](#)
- [DigitalOcean exporter](#)
- [Docker Cloud exporter](#)
- [Docker Hub exporter](#)
- [GitHub exporter](#)
- [InstaClustr exporter](#)
- [Mozilla Observatory exporter](#)
- [OpenWeatherMap exporter](#)
- [Pagespeed exporter](#)
- [Rancher exporter](#)
- [Speedtest exporter](#)

### Other monitoring systems

- [Akamai Cloudmonitor exporter](#)
- [AWS CloudWatch exporter](#) (official)
- [Cloud Foundry Firehose exporter](#)
- [Collectd exporter](#) (official)
- [Google Stackdriver exporter](#)
- [Graphite exporter](#) (official)
- [Heka dashboard exporter](#)
- [Heka exporter](#)
- [InfluxDB exporter](#) (official)
- [JavaMelody exporter](#)
- [JMX exporter](#) (official)
- [Munin exporter](#)
- [Nagios / Naemon exporter](#)

### New Relic exporter

- [NRPE exporter](#)
- [Osquery exporter](#)
- [Pingdom exporter](#)
- [scollector exporter](#)
- [Sensu exporter](#)
- [SNMP exporter](#) (official)
- [StatsD exporter](#) (official)

### Miscellaneous

- [BIG-IP exporter](#)
- [BIND exporter](#)
- [Bitbucket exporter](#)
- [Blackbox exporter](#) (official)
- [BOSH exporter](#)
- [cAdvisor](#)
- [Confluence exporter](#)
- [Dovecot exporter](#)
- [Jenkins exporter](#)
- [JIRA exporter](#)
- [Kannel exporter](#)
- [Kemp LoadBalancer exporter](#)
- [Meteor JS web framework exporter](#)
- [Minecraft exporter mod](#)
- [PHP-FPM exporter](#)
- [PowerDNS exporter](#)
- [Process exporter](#)
- [rTorrent exporter](#)
- [SABnzbd exporter](#)
- [Script exporter](#)

<https://prometheus.io/docs/instrumenting/exporters/>

## 二、传统VM的系统监控 2.6 应用业务指标监控--springboot

### 1、应用程序配置

- 添加监控注解：@EnablePrometheusEndpoint
- 添加监控注解：@EnableSpringBootMetricsCollector

### 2、业务逻辑中添加指标监控数据

- 每次http请求随机造一些成功、失败计数
- 将构建的jar包放到服务器上启动起来

### 3、配置Prometheus

```
- job_name: 'java_metrics'
  metrics_path: '/prometheus'
  static_configs:
    - targets: ['10.20.26.139:8888']
```

### 4、Prometheus抓取的指标数据:http://host:9090/graph; 查询指标 “my\_sample\_counter”

Element

```
my_sample_counter{instance="10.20.26.139:8888",job="java_metrics",status="error"}
```

```
my_sample_counter{instance="10.20.26.139:8888",job="java_metrics",status="success"}
```

### 5、Grafana创建监控面板：配置模板变量，按节点监控

```
@SpringBootApplication
@EnablePrometheusEndpoint
@EnableSpringBootMetricsCollector
public class TestMetricsApplication {

    public static void main(String[] args) {
        SpringApplication.run(TestMetricsApplication.class, args);
    }

    private static Random random = new Random();

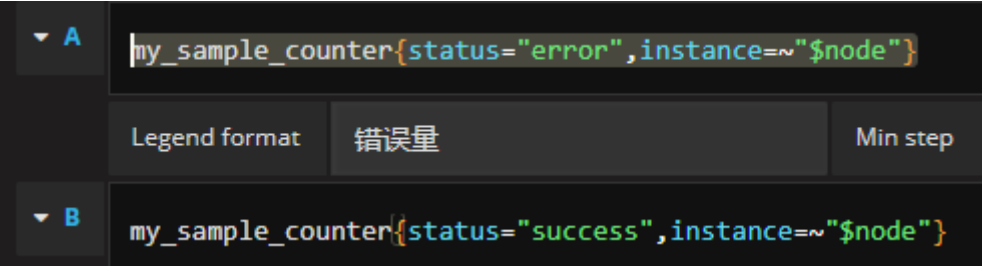
    private static final Counter requestTotal = Counter.build()
        .name("my_sample_counter")
        .labelNames("status")
        .help("A simple Counter to illustrate custom Counters in :")

    @RequestMapping("/endpoint")
    public void endpoint() {
        if (random.nextInt(2) > 0) {
            requestTotal.labels("success").inc();
        } else {
            requestTotal.labels("error").inc();
        }
    }
}
```

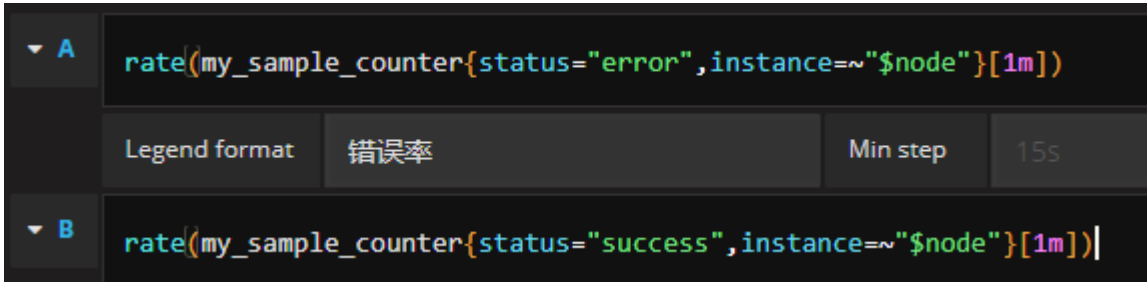
```
$node label_values(my_sample_counter{job="java_metrics"}, instance)
```

# 二、传统VM的系统监控 2.7应用业务指标监控--springboot

## 1. Grafana新建监控面板：请求数量监控配置



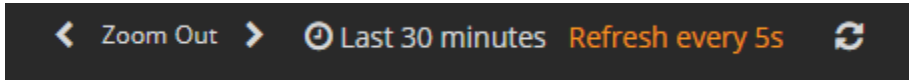
## 2. Grafana新建监控面板：每分钟成功、错误率



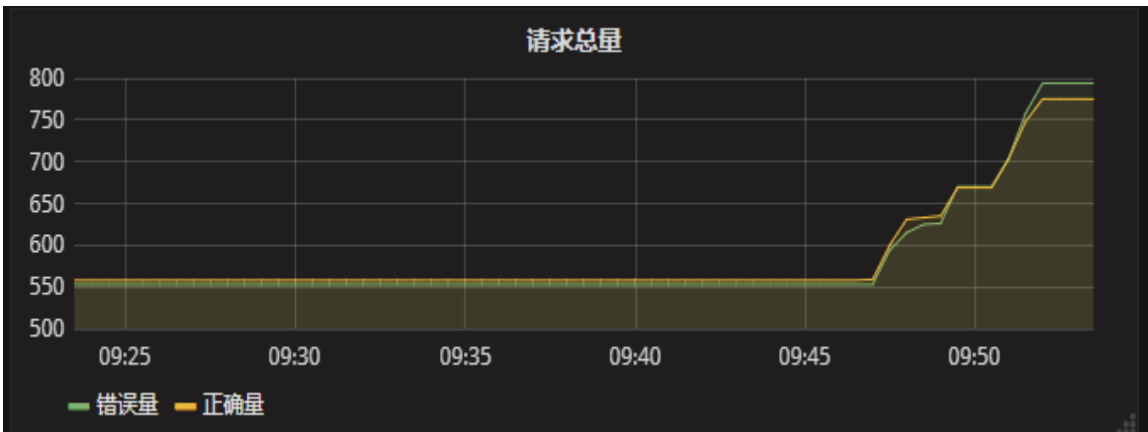
应用监控面板

## 3. 浏览器不断访问应用的服务接口：<http://10.20.26.139:8888/endpoint>

## 4. Grafana配置查看最近30分钟监控，每5秒钟自动刷新:



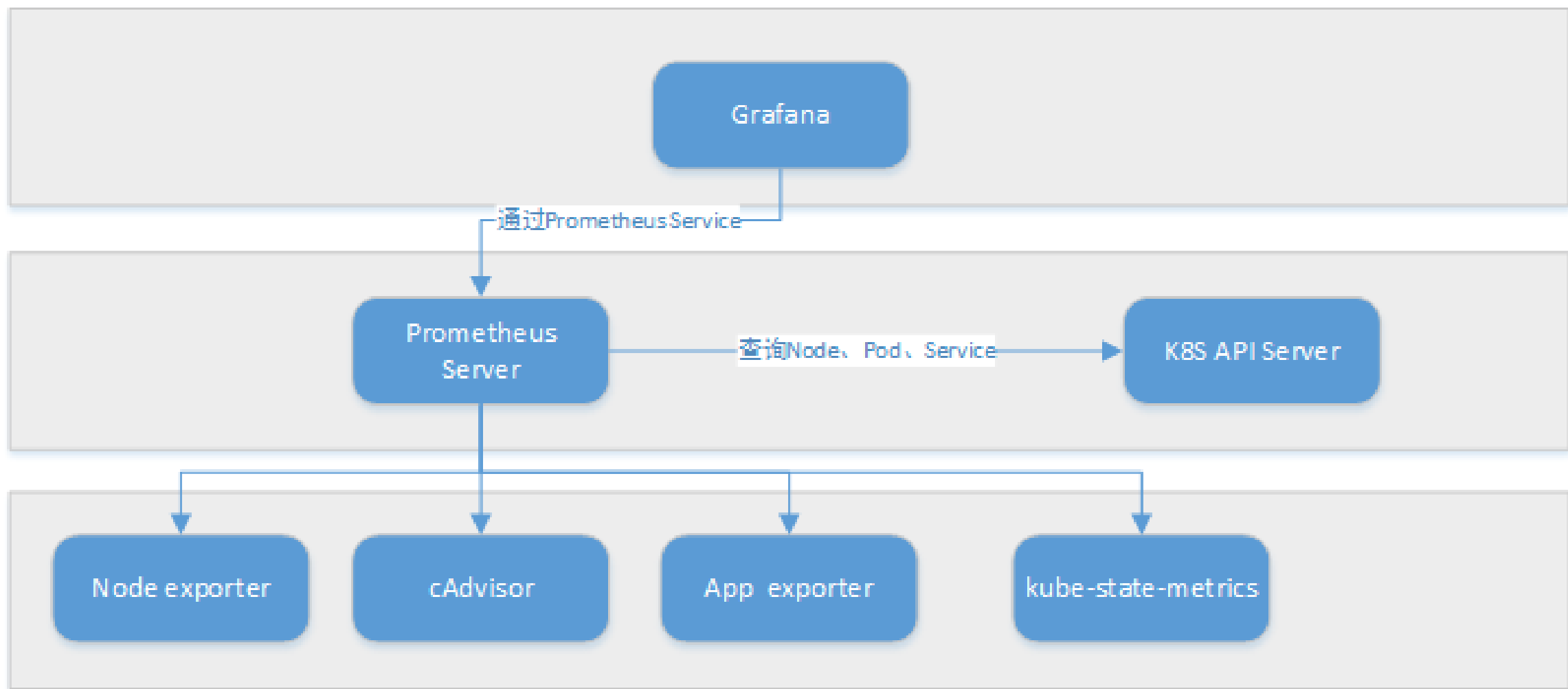
## 5. Grafana 应用指标：请求数量监控展示



## 6. Grafana 应用指标：每分钟流量监控



### 三、基于K8S的容器监控 3.1 K8S环境中的系统监控架构图



#### 提示：

1. 因为用到了基于K8S的自动发现节点、POD以及service；
2. 需要事先确定命名空间是否有权限创建role以及cluster role；如果没有权限需要开通相应的权限

# 三、基于K8S的容器监控

## 3.2 K8S内Prometheus环境搭建

### 1、安装参考材料

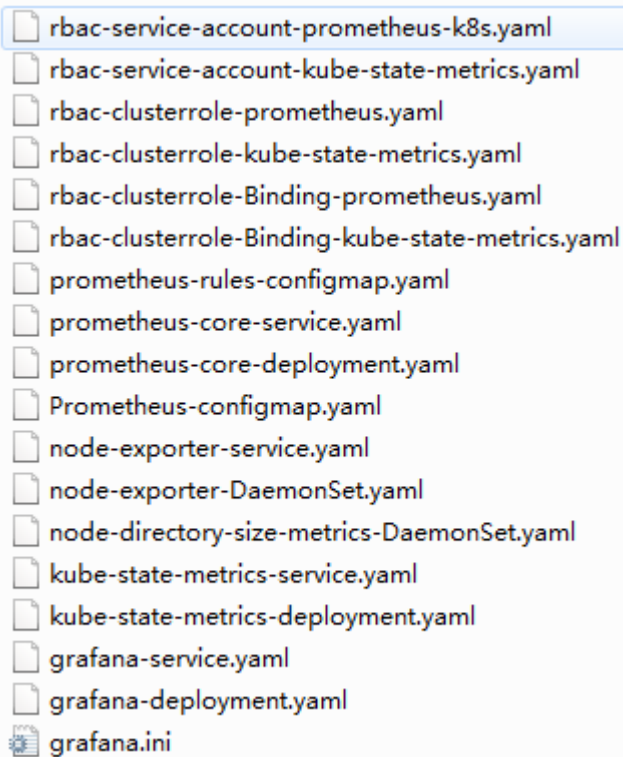
- <http://blog.csdn.net/wenwst/article/details/76624019>

### 2、安装Prometheus

- 命名空间使用已有：szd-0879950f
- 准备好文档中需要的镜像
- 创建rbac资源：变更文档中的创建顺序，先创建rbac资源
  - 创建ServiceAccount：服务账号
  - 创建ClusterRole：集群角色，定义能够访问的资源
  - 创建ClusterRoleBinding：将服务账号与集群角色绑定
- 部署node-exporter及服务
- 部署kube-state-metrics及服务
- 部署node-directory-size-metrics及服务
- 部署Prometheus及服务和configmap
- 数据库使用已有的数据库服务
- 部署grafana及服务及configmap

通过yaml创建资源及服务：

```
kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f apply -f xxx.yaml
```



- rbac-service-account-prometheus-k8s.yaml
- rbac-service-account-kube-state-metrics.yaml
- rbac-clusterrole-prometheus.yaml
- rbac-clusterrole-kube-state-metrics.yaml
- rbac-clusterrole-Binding-prometheus.yaml
- rbac-clusterrole-Binding-kube-state-metrics.yaml
- prometheus-rules-configmap.yaml
- prometheus-core-service.yaml
- prometheus-core-deployment.yaml
- Prometheus-configmap.yaml
- node-exporter-service.yaml
- node-exporter-DaemonSet.yaml
- node-directory-size-metrics-DaemonSet.yaml
- kube-state-metrics-service.yaml
- kube-state-metrics-deployment.yaml
- grafana-service.yaml
- grafana-deployment.yaml
- grafana.ini

安装涉及的yaml定义文件：



k8s-yaml.zip

**提示：**

**1. 按照上面安装参考资料安装后，容器监控获取不到数据，后面详细介绍解决方法；**

```
- job_name: 'kubernetes-endpoints'
  kubernetes_sd_configs:
    - role: endpoints
  relabel_configs:
    - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_scrape]
      action: keep
      regex: true
    - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_scheme]
      action: replace
      target_label: __scheme__
      regex: (https?)
    - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_path]
      action: replace
      target_label: __metrics_path__
      regex: (.+)
    - source_labels: [__address__, __meta_kubernetes_service_annotation_prometheus_io_port]
      action: replace
      target_label: __address__
      regex: (.+)(?::\d+);(\d+)
      replacement: $1:$2
    - action: labelmap
      regex: __meta_kubernetes_service_label_(.+)
    - source_labels: [__meta_kubernetes_namespace]
      action: replace
      target_label: kubernetes_namespace
    - source_labels: [__meta_kubernetes_service_name]
      action: replace
      target_label: kubernetes_name
```

通过K8S获取所有endpoint  
通过默认的metrics获取指标

对默认的标签做重写

获取Prometheus的服务端口

```
[root@SZD-L0077282 ~]# kubectl -s 10.25.65.209:8080 --namespace=szd-0879950f get svc/prometheus
prometheus 172.254.226.53 <nodes> 9090:30399/TCP 2d
prometheus-node-exporter None <none> 9100/TCP 2d
```

查看监控目标：<http://10.25.85.46:30399/targets>

kubernetes-endpoints				
Endpoint	State	Labels	Last Scrape	Error
http://10.25.65.28:9100/metrics	DOWN	app="prometheus" component="node-exporter" instance="10.25.65.28:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	17.404s ago	context deadline exceeded
http://10.25.65.29:9100/metrics	DOWN	app="prometheus" component="node-exporter" instance="10.25.65.29:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	17.405s ago	context deadline exceeded
http://10.25.80.178:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.80.178:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	1.659s ago	
http://10.25.80.179:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.80.179:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	2.157s ago	
http://10.25.81.138:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.81.138:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	3.725s ago	
http://10.25.81.169:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.81.169:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	9.77s ago	
http://10.25.81.177:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.81.177:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	370ms ago	
http://10.25.81.204:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.81.204:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	3.613s ago	
http://10.25.81.205:9100/metrics	UP	app="prometheus" component="node-exporter" instance="10.25.81.205:9100" kubernetes_name="prometheus-node-exporter" kubernetes_namespace="szd-0879950f"	3.001s ago	



# 三、基于K8S的容器监控 3.4 K8S节点基础信息监控—监控效果

## 1、通过 Prometheus查看指标

- <http://10.25.85.46:30399/graph>
- Node开头的指标都是节点指标

## 2、Grafana导入监控面板模板

- Grafana->Dashboards->import
- 导入后即可实现对数据指标的监控

## 2、监控面板修改

- 默认的监控是所有节点的数据加和
- 新增节点变量



k8snode.json

```
$Node label_values(node_boot_time{job="kubernetes-endpoints"}, instance)
```

- 指标查询语言中添加基于节点的数据过滤

```
Query sum(rate(node_cpu{mode="idle",instance=~"~^$Node$"}[2m])) * 100
```

- 最终实现可以查看单节点指标也可以查看所有节点指标和

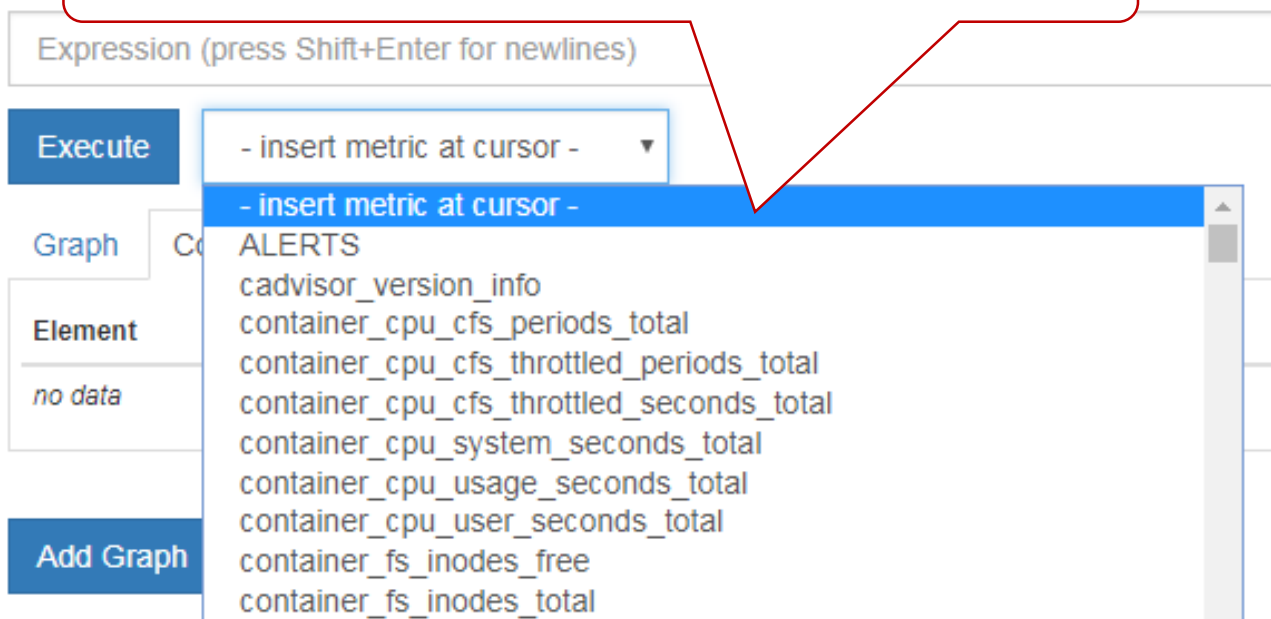


```
- job_name: 'kubernetes-containers'
  tls_config:
    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
    bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
  kubernetes_sd_configs:
    - role: node
  relabel_configs:
    - source_labels: [__address__]
      regex: '(.*):10250'
      replacement: '${1}:4194'
      target_label: __address__
    - source_labels: [__meta_kubernetes_node_label_kubernetes_io_hostname]
      target_label: kubernetes_io_hostname
```

提示：

1. Grafana官方网站获取到的k8s容器监控面板都是基于cAdvisor生成的指标数据，环境安装完成后获取不到容器相关指标数据
2. 确定cAdvisor服务正常后添加左边的配置，通过nodeip:4194/metrics 获取容器指标
3. Grafana通过kubernetes\_io\_hostname获取主机ip，所以对原有标签进行了修改

新增配置后Prometheus能看到容器指标



新增配置后Prometheus可以看到监控目标

Targets

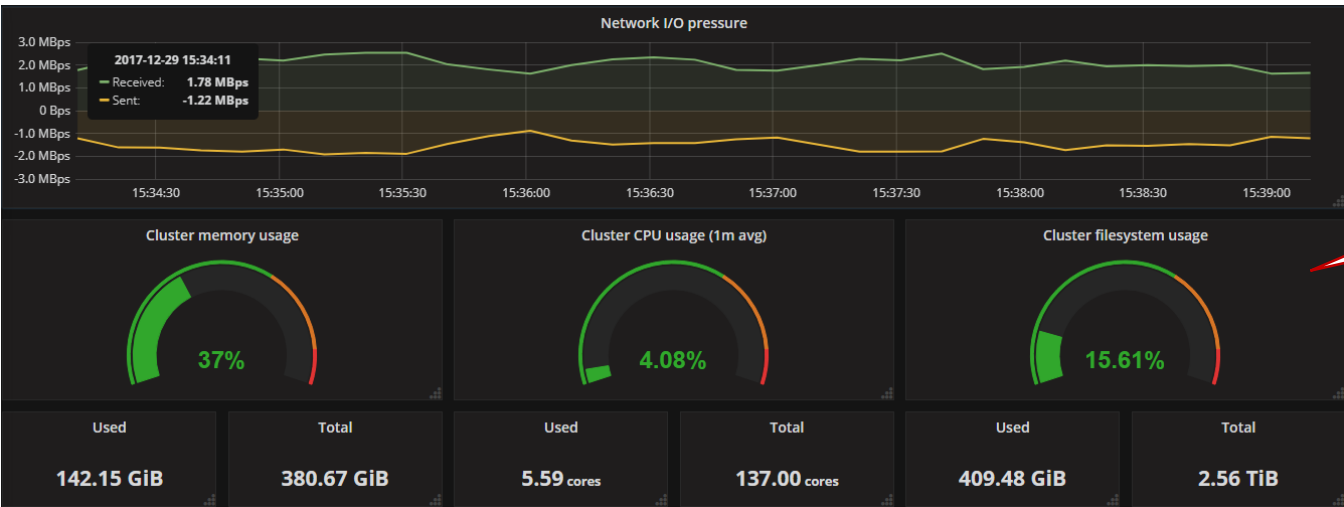
kubernetes-containers

Endpoint	State	Labels
http://10.25.65.28:4194/metrics	DOWN	instance="10.25.65.28" kubernetes_io_hostname="10.25.65.28"
http://10.25.65.29:4194/metrics	DOWN	instance="10.25.65.29" kubernetes_io_hostname="10.25.65.29"
http://10.25.80.178:4194/metrics	UP	instance="10.25.80.178" kubernetes_io_hostname="10.25.80.178"
http://10.25.80.179:4194/metrics	UP	instance="10.25.80.179" kubernetes_io_hostname="10.25.80.179"
http://10.25.81.138:4194/metrics	UP	instance="10.25.81.138" kubernetes_io_hostname="10.25.81.138"
http://10.25.81.169:4194/metrics	UP	instance="10.25.81.169" kubernetes_io_hostname="10.25.81.169"



三、基于K8S的容器监控

3.5 K8S 容器信息监控—效果展示



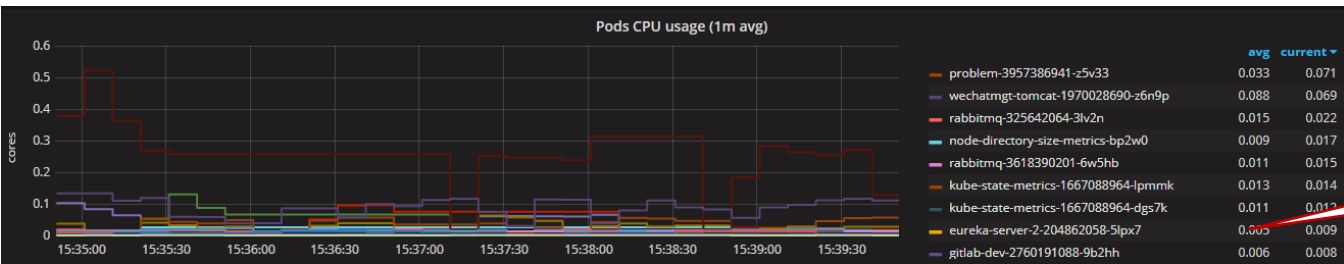
网络及CPU汇总监控



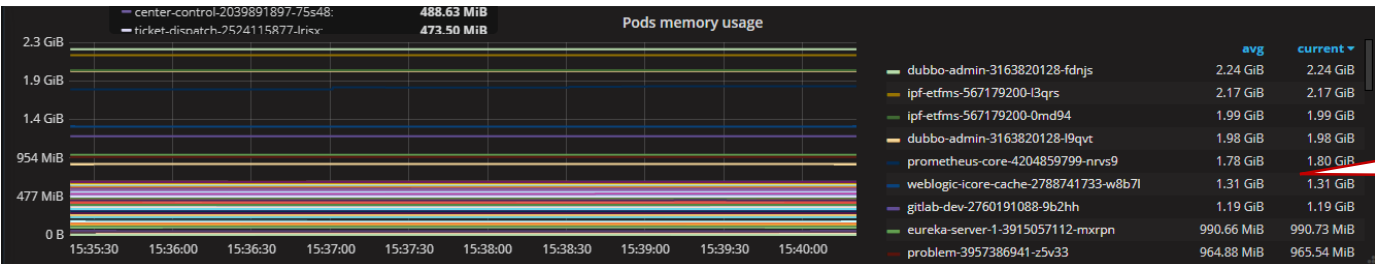
监控面板模板



pod-monitor.png



POD CPU监控



POD 内存监控

# 三、基于K8S的容器监控

## 3.6 K8S内Prometheus监控非容器环境服务器基础参数

```
- job_name: 'external_node'
  static_configs:
    - targets: ['10.20.26.139:9100', '10.20.26.140:9100']
      labels:
        group: 'monitor'
```

添加容器环境外服务器节点

Grafana导入面板后的监控效果

Prometheus出现监控目标



external_node		
Endpoint	State	Labels
http://10.20.26.139:9100/metrics	UP	group="monitor" instance="10.20.26.139:9100"
http://10.20.26.140:9100/metrics	UP	group="monitor" instance="10.20.26.140:9100"

三、基于K8S的容器监控

3.7 K8S内Prometheus监控非容器环境中间件服务

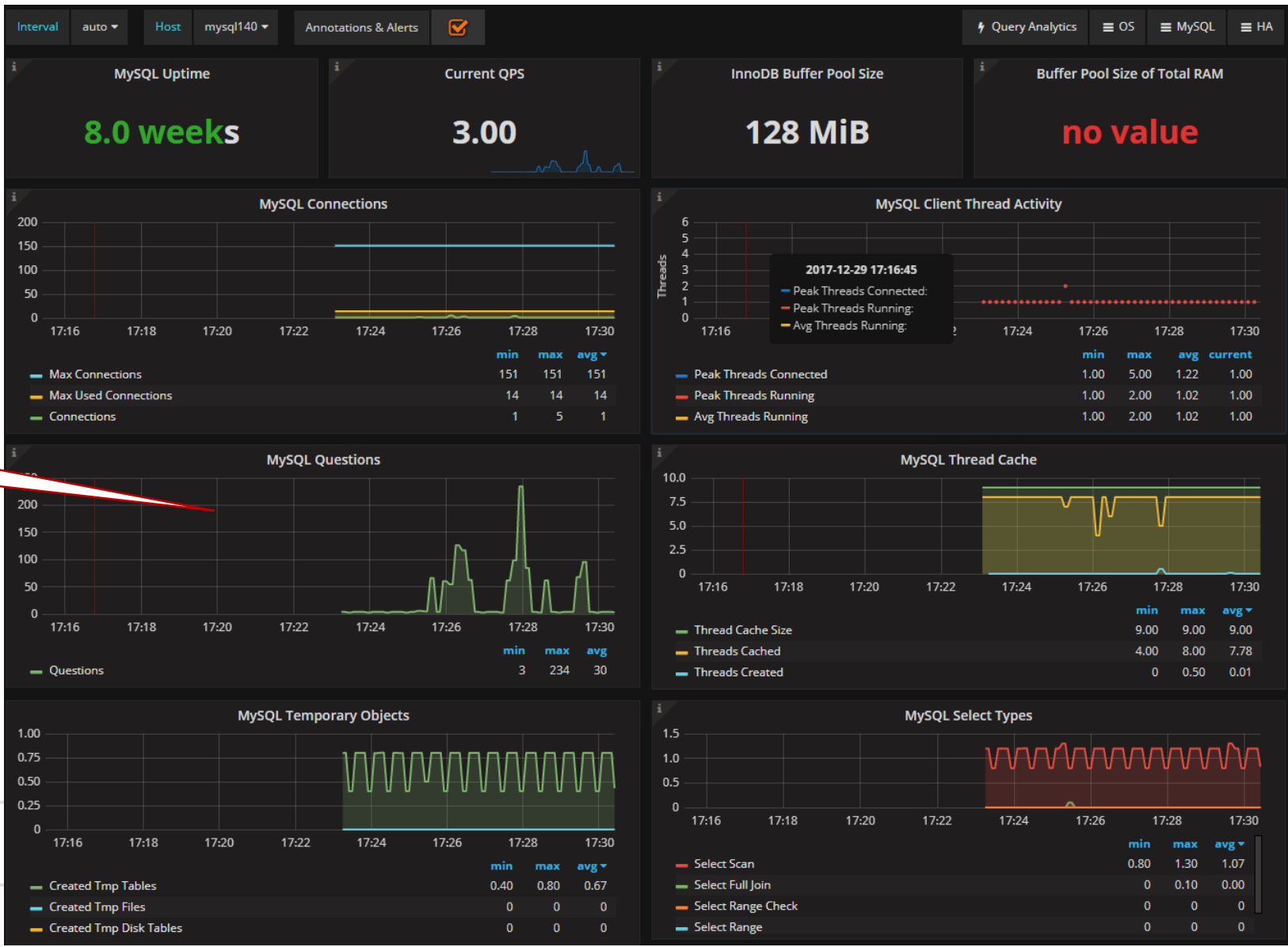
```
- job_name: 'mysql'
static_configs:
- targets: ['10.20.26.140:9104']
labels:
instance: 'mysql140'
```

添加容器环境外数据库服务

Grafana监控效果图

Prometheus出现监控目标

mysql		
Endpoint	State	Labels
http://10.20.26.140:9104/metrics	UP	instance="mysql140"



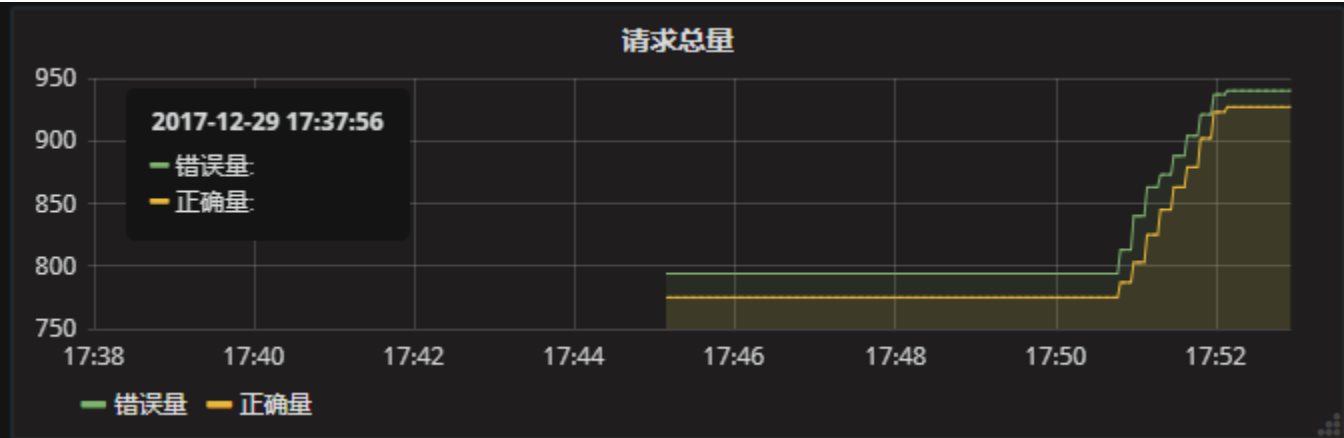
```
- job_name: 'java_metrics'  
  metrics_path: '/prometheus'  
  static_configs:  
    - targets: ['10.20.26.139:8888']
```

添加容器环境外应用服务

Grafana监控效果图

Prometheus出现监控目标

java_metrics		
Endpoint	State	Labels
http://10.20.26.139:8888/prometheus	UP	instance="10.20.26.139:8888"



四、容器内应用监控

4.2 K8S内Prometheus监控容器应用——构建容器服务

服务信息

基础信息

服务名 javametrics

服务状态 创建完成

所属应用 javametrics

所属集群 容器测试

区域 SZD

创建人 zhaoyanbin442

创建时间 2018-01-03 10:07:35

访问模式 集群外访问

服务IP 172.254.220.26

容器组	事件	数据卷	端口
负载均衡			
容器组名称	运行状态	ip	宿主机IP
javametrics-678295983-6t6ms	● 运行中	172.1.137.5	10.25.85.46

通过caas拉起应用服务

以jdk8基础镜像通过docerfile  
构建示例应用镜像

编写launch.sh 启动脚本

配置服务负载均衡入口以及测试服务请求

```
# Pull base image
# -----
FROM hub.yun.paic.com.cn/official/jdk:8

# Maintainer
# -----
MAINTAINER zhaoyanbin<zhaoyanbin442>

RUN mkdir -p /opt/javametrics/

COPY TestMetrics-0.0.1-SNAPSHOT.jar launch.sh /opt/javametrics/

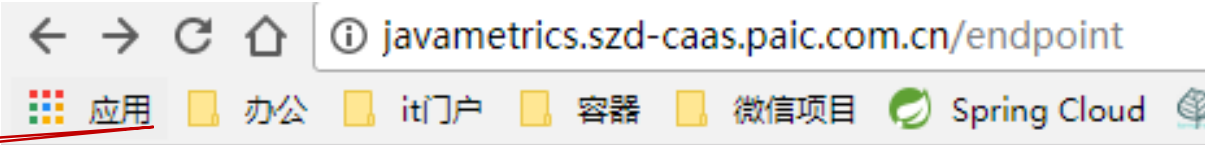
# Install and configure Oracle JDK
# -----
WORKDIR /opt/javametrics/

CMD [ "bash" , "launch.sh" ]
```

```
#!/bin/sh

java -jar TestMetrics-0.0.1-SNAPSHOT.jar
```

名称	类型	负载服务	服务端口	域名
<input type="checkbox"/> javametrics	Nginx	javametrics	8888	javametrics.szd-caas.paic.com.cn



success

### 1、修改caas生成的service添加注解以及标签

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: 2018-01-03T02:08:28Z
  annotations:
    prometheus.io/scrape: 'true'
    prometheus.io/path: '/prometheus'
  labels:
    app: javametrics
name: javametrics
```

### 2、修改原来的Prometheus配置，注销之前的虚拟机javametrics监控配置

```
#- job_name: 'java_metrics'
# metrics_path: '/prometheus'
# static_configs:
#   - targets: ['10.20.26.139:8888']
```

3、刷新Prometheus targets 自动发现被监控服务

http://172.1.137.5:8888/prometheus	UP	app="javametrics" instance="172.1.137.5:8888" kubernetes_name="javametrics" kubernetes_namespace="szd-0879950f"
------------------------------------	----	---

### 4、查询监控指标，可见监控数据已经是来自新的容器服务

my\_sample\_counter

Execute

my\_sample\_counter

Graph

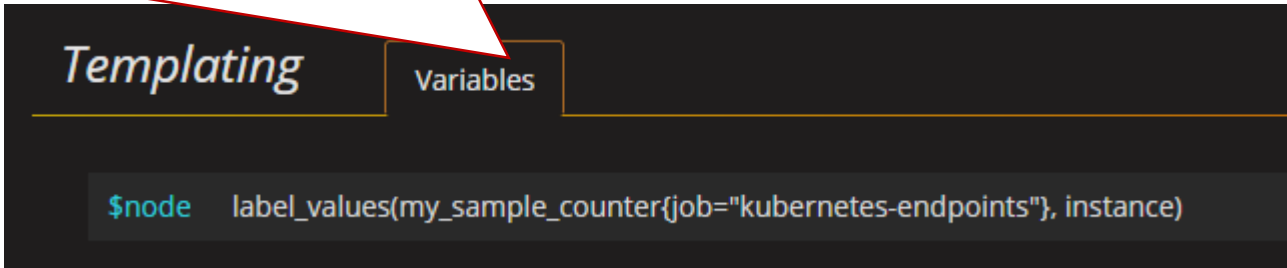
Console

Element

my\_sample\_counter{app="javametrics",instance="172.1.137.5:8888",job="kubernetes-endpoints",kubernetes\_name="javametrics",kubernetes\_namespace="szd-0879950f",status="error"}

my\_sample\_counter{app="javametrics",instance="172.1.137.5:8888",job="kubernetes-endpoints",kubernetes\_name="javametrics",kubernetes\_namespace="szd-0879950f",status="success"}

1、修改Grafana模板变量，更改对应的job标签值



2、每分钟请求  
监控



3、请求总量监控



可见：

- 1. 传统应用在没有第三方工具比如etcd等支持的情况下，如果想新增监控必须要修改Prometheus配置文件
- 2. 基于K8S的Prometheus可以实现监控自动发现，遵循一定规则下不需要进行任何配置就可以发现新的监控目标并进行展示

## 五、参考资料

### 参考资料：

Prometheus官方资料：<https://prometheus.io/docs/instrumenting/exporters/>

Grafana官方资料：<https://grafana.com/dashboards>

虚拟机环境安装参考：<http://blog.51cto.com/youerning/2050543?from=timeline>

容器环境安装参考：<http://blog.csdn.net/wenwst/article/details/76624019>

SpringBoot监控：<https://yq.aliyun.com/articles/272542>

材料中涉及到的测试程序及脚本：<http://10.11.112.66/docker-container-scripts/prometheus-monitor>



同呼吸、共命运、心连心

中国平安 PINGAN

保险 · 银行 · 投资