# 监控Prometheus集成文档

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **主要更改内容** | **更改人** | **更改日期** |
| 1 | **初始文档** | **韩邦雪** | **2018-11-08** |

目录

[简介](#_Toc1187_WPSOffice_Level1) [1](#_Toc1187_WPSOffice_Level1)

[集成](#_Toc27188_WPSOffice_Level1) [1](#_Toc27188_WPSOffice_Level1)

[1、 环境要求](#_Toc13954_WPSOffice_Level1) [1](#_Toc13954_WPSOffice_Level1)

[2、 配置要求](#_Toc17741_WPSOffice_Level1) [1](#_Toc17741_WPSOffice_Level1)

[3、 验证配置](#_Toc891_WPSOffice_Level1) [4](#_Toc891_WPSOffice_Level1)

## 简介

Prometheus 是一套开源的系统监控报警框架。它启发于 Google 的 borgmon 监控系统，由工作在 SoundCloud 的 google 前员工在 2012 年创建，作为社区开源项目进行开发，并于 2015 年正式发布。2016 年，Prometheus 正式加入 Cloud Native Computing Foundation，成为受欢迎度仅次于 Kubernetes 的项目。

## 集成

### 环境要求

① springboot2.0+

### 配置要求

① 在项目中引入maven配置

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

<!-- springboot内置收集系统数据功能包 -->

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-aop</artifactId>

<!-- springboot aop功能包 -->

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-registry-prometheus</artifactId>

<!-- prometheus收集数据功能包 -->

</dependency>

② 在项目yml或properties配置中暴露相关endpoint

management.metrics.export.prometheus.enabled=true

management.metrics.export.prometheus.descriptions=true

management.endpoints.web.exposure.include=\*

③ 如果有需要监控service层 dao层方法执行相关信息请添加如下代码，**不添加默认提供接口调用相关信息**。粒度相对service dao层大一些

**package** com.yooli.consul.client.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.concurrent.TimeUnit;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Pointcut;

**import** org.aspectj.lang.reflect.MethodSignature;

**import** org.springframework.stereotype.Component;

**import** io.micrometer.core.instrument.ImmutableTag;

**import** io.micrometer.core.instrument.Metrics;

**import** io.micrometer.core.instrument.Tag;

**import** io.micrometer.core.instrument.Timer;

@Aspect

@Component

**public** **class** PrometheusAop {

List<Tag> init(String methodName) {

ArrayList<Tag> list = **new** ArrayList<Tag>();

list.add(**new** ImmutableTag("methodName", methodName));

**return** list;

}

List<Tag> exception(String methodName) {

ArrayList<Tag> list = **new** ArrayList<Tag>();

list.add(**new** ImmutableTag("methodName", methodName+"->Exception"));

**return** list;

}

/\*\*

\* 指定切点

\*/

@Pointcut("execution(public \* com.yooli.consul.client.controller.\*.\*(..))")

**public** **void** webLog1() {

}

@Pointcut("execution(public \* com.yooli.carloan.batch.\*.\*(..))")

**public** **void** webLog2() {

}

@Pointcut("execution(public \* com.yooli.cbs.api.\*.\*(..))")

**public** **void** webLog3() {

}

@Pointcut("execution(public \* com.yooli.carloan.web.test..\*.\*(..))")

**public** **void** webLog4() {

}

/\*\*

\* 环绕通知,环绕增强，相当于MethodInterceptor

\*

\* **@param** proceedingJoinPoint

\* proceedingJoinPoint

\* **@return** object

\* **@throws** Throwable

\*/

@Around("webLog1()||webLog2()||webLog3()||webLog4()")

**public** Object around(ProceedingJoinPoint proceedingJoinPoint) **throws** Throwable {

MethodSignature signature = (MethodSignature) proceedingJoinPoint.getSignature();

String methodName = signature.getDeclaringTypeName() + "." + signature.getName();

**long** startTime = System.*currentTimeMillis*();

Object object;

**try** {

object = proceedingJoinPoint.proceed();

Timer timer = Metrics.*timer*("service", init(methodName));

timer.record(System.*currentTimeMillis*() - startTime, TimeUnit.***MILLISECONDS***);

} **catch** (Throwable e) {

Timer exceptionTimer = Metrics.*timer*("service", exception(methodName));

exceptionTimer.record(System.*currentTimeMillis*() - startTime, TimeUnit.***MILLISECONDS***);

**throw** e;

}

**return** object;

}

}

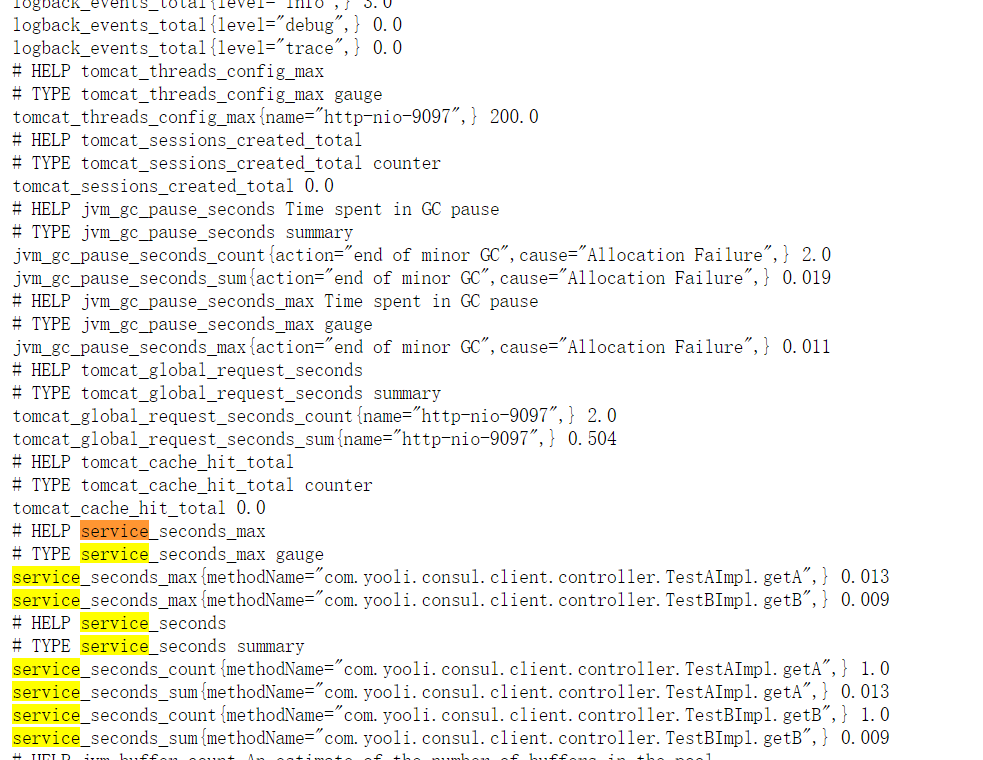
在自己本项目中任意可以被扫描到的地方添加该类，需要自己配置aop pointcut如"execution(public \* com.yooli.cbs.api.\*.\*(..))" 指定需要切入的方法即可。

④ 线上服务器需添加node\_exporter收集服务器信息

⑤ 找运维或者架构组成员开通账号即可

### 3、验证配置

访问localhost:port/actuator/prometheus 如出现如下数据则成功



如果配置了aop的可以调用切面配置的方法再访问localhost:port/actuator/prometheus 如果在返回值出现方法信息则表示成功 如上图的黄线行