

第十三章 微服务监控

一样的在线教育，不一样的教学品质



目录 Contents

- ◆ Prometheus 监控实践
- ◆ Spring Boot Admin 监控实践
- ◆ JVM GC 日志可视化分析

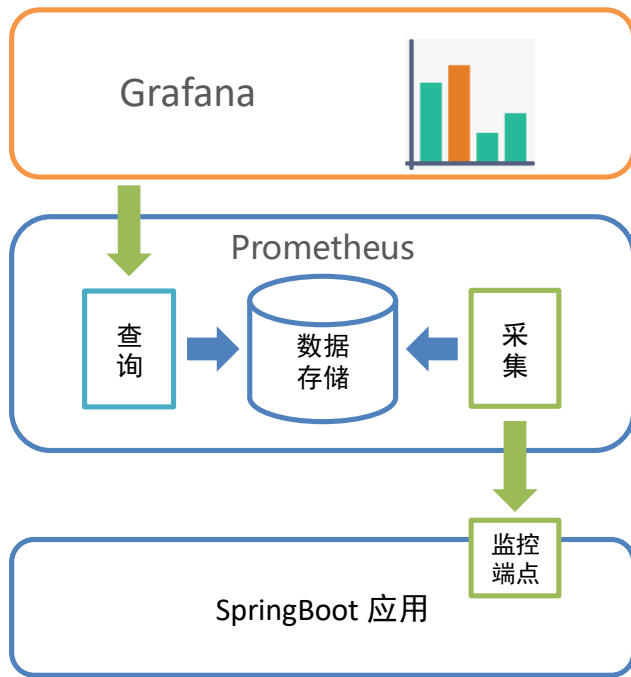
小节导学

Prometheus (普罗米修斯) 是一个开源的系统监控与告警工具，用于收集、存储监控数据，并提供数据查询服务。

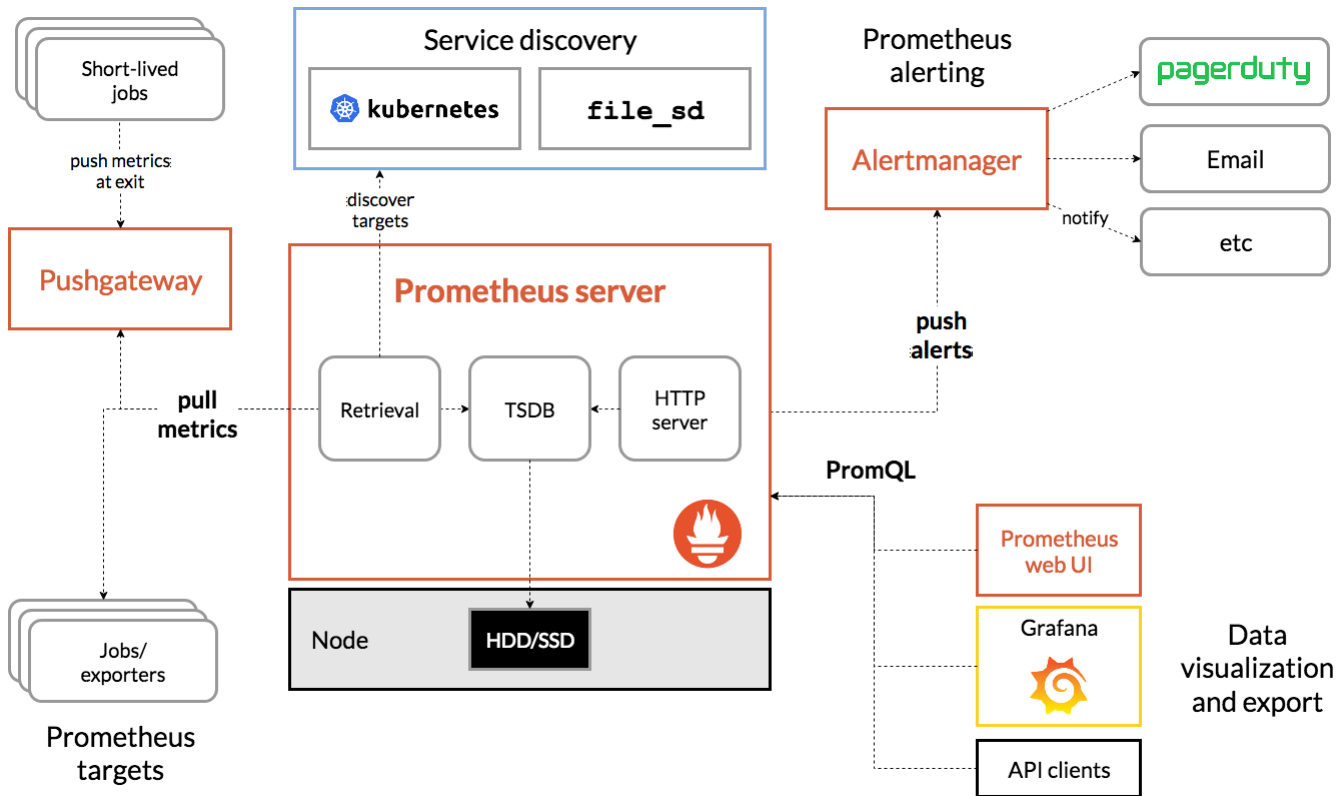
SpringBoot 的 actuator 提供了监控端点，SpringBoot2 中引入了 **micrometer**，可以更方便的对接各种监控系统，包括 Prometheus。

Grafana 是专业 UI 仪表盘系统，支持 Prometheus 数据源。

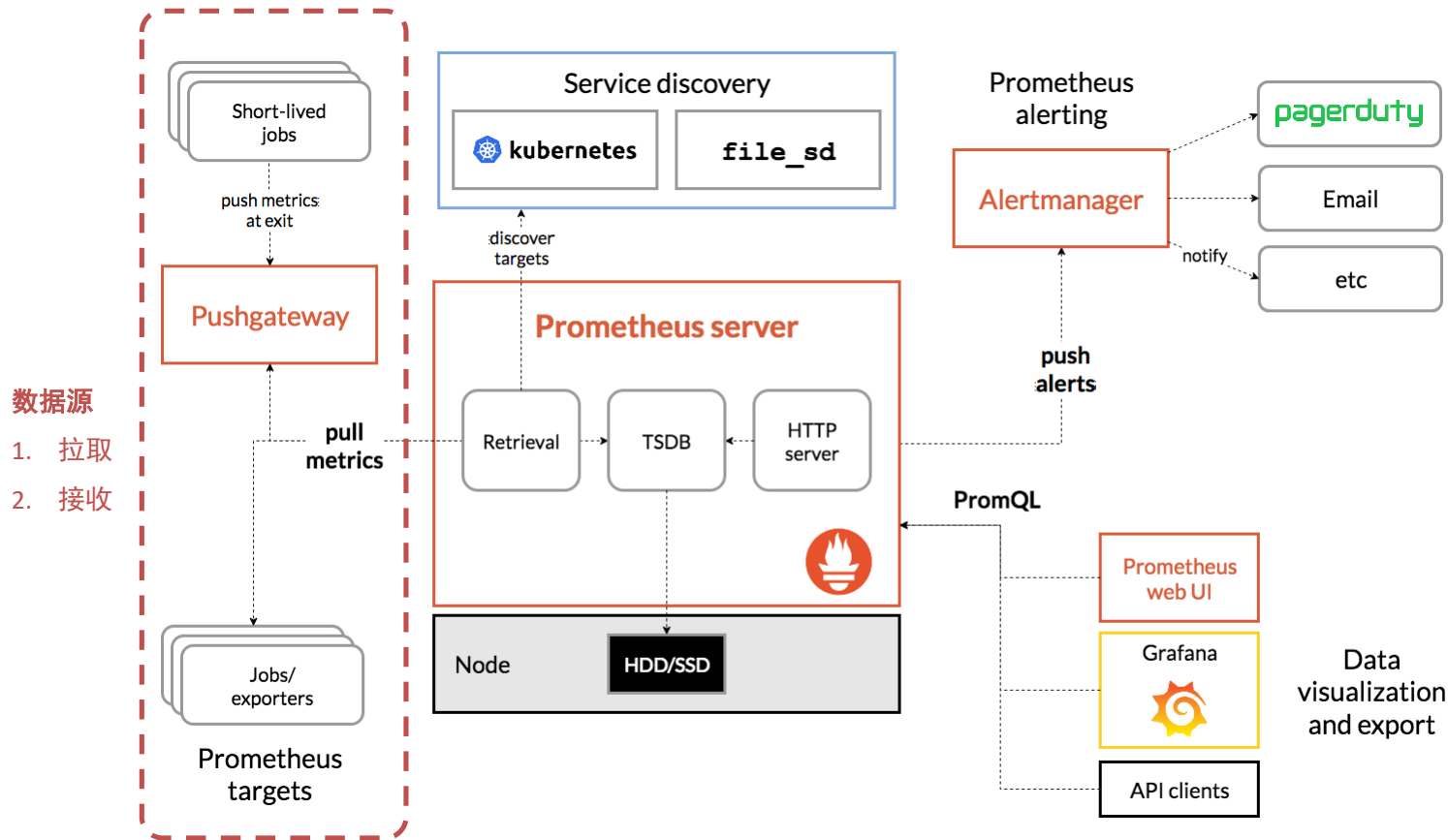
- Prometheus 架构
- Prometheus + Grafana 监控 SpringBoot 实践



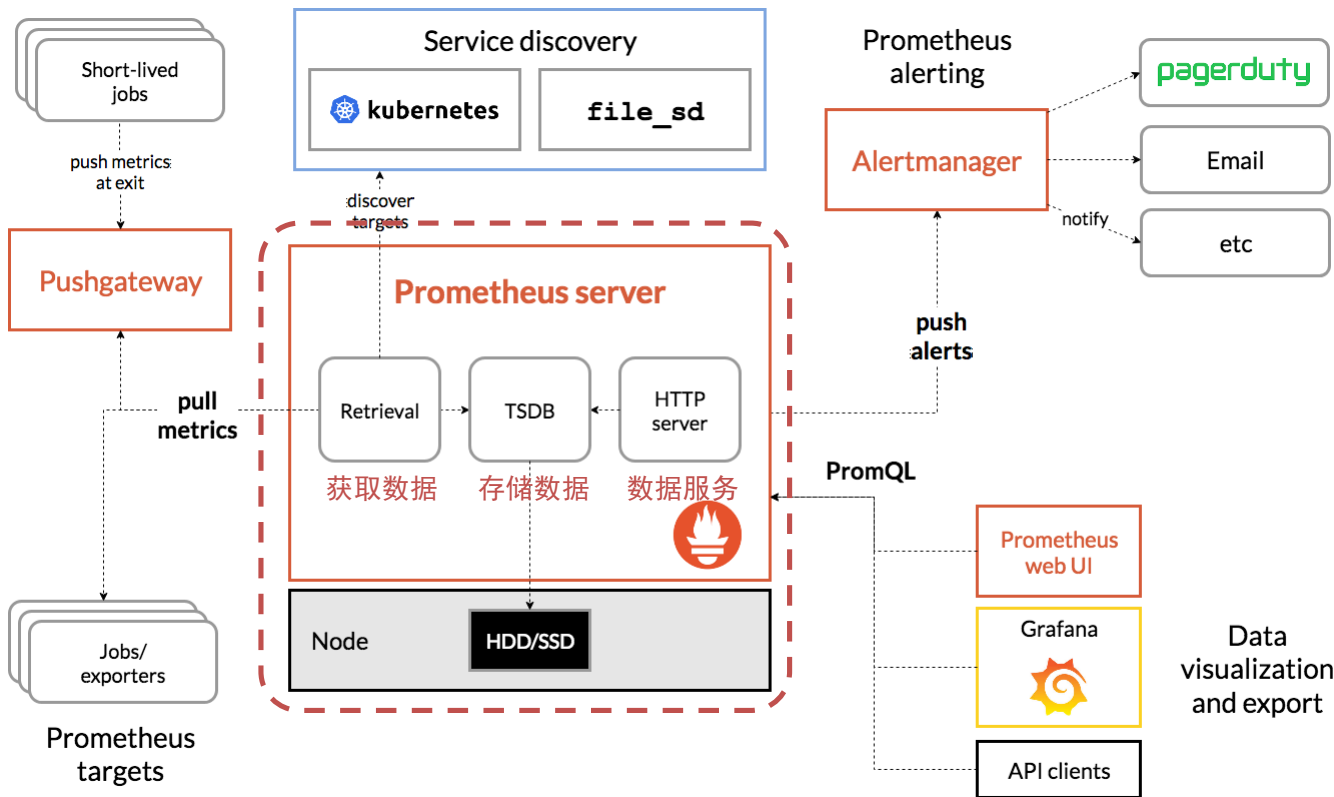
1. Prometheus 架构



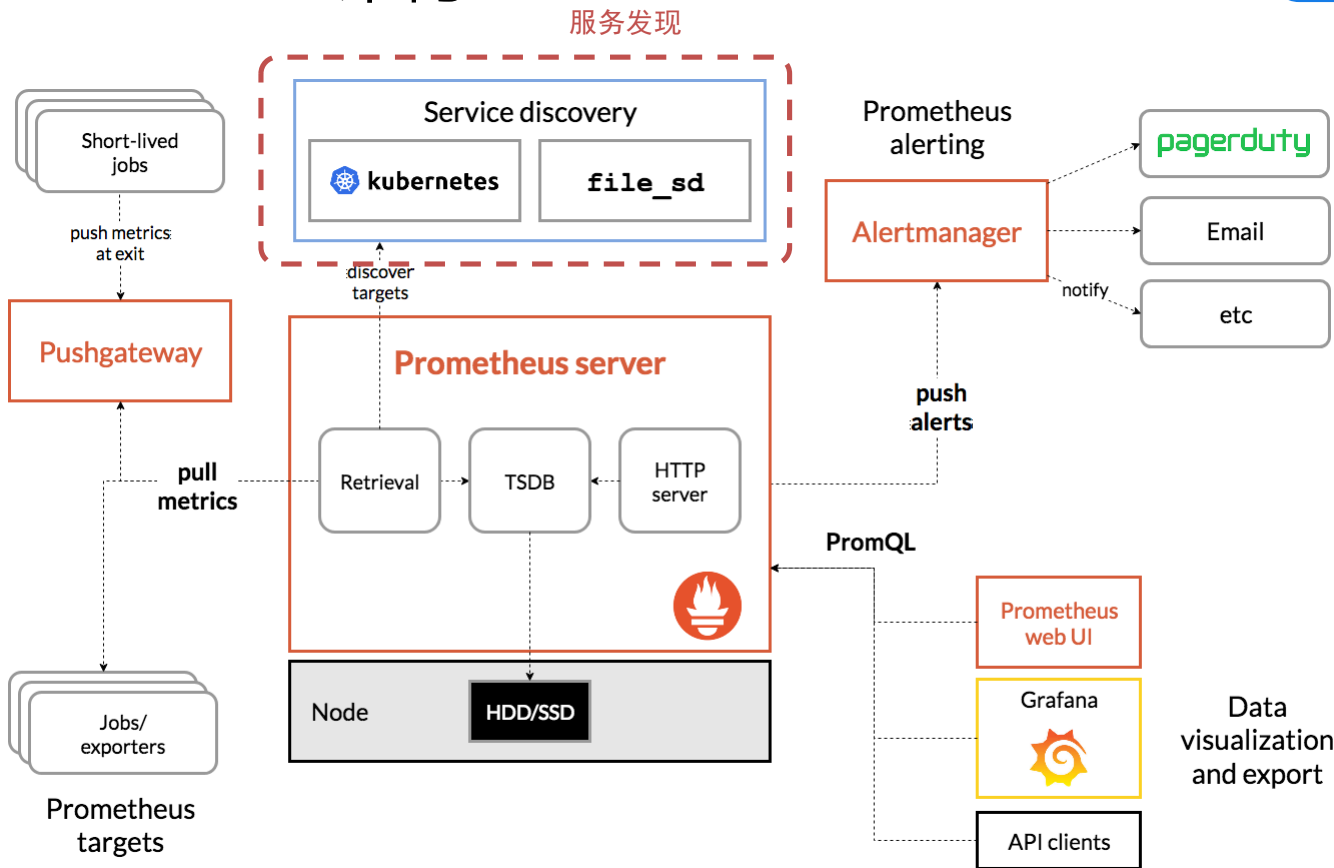
1. Prometheus 架构



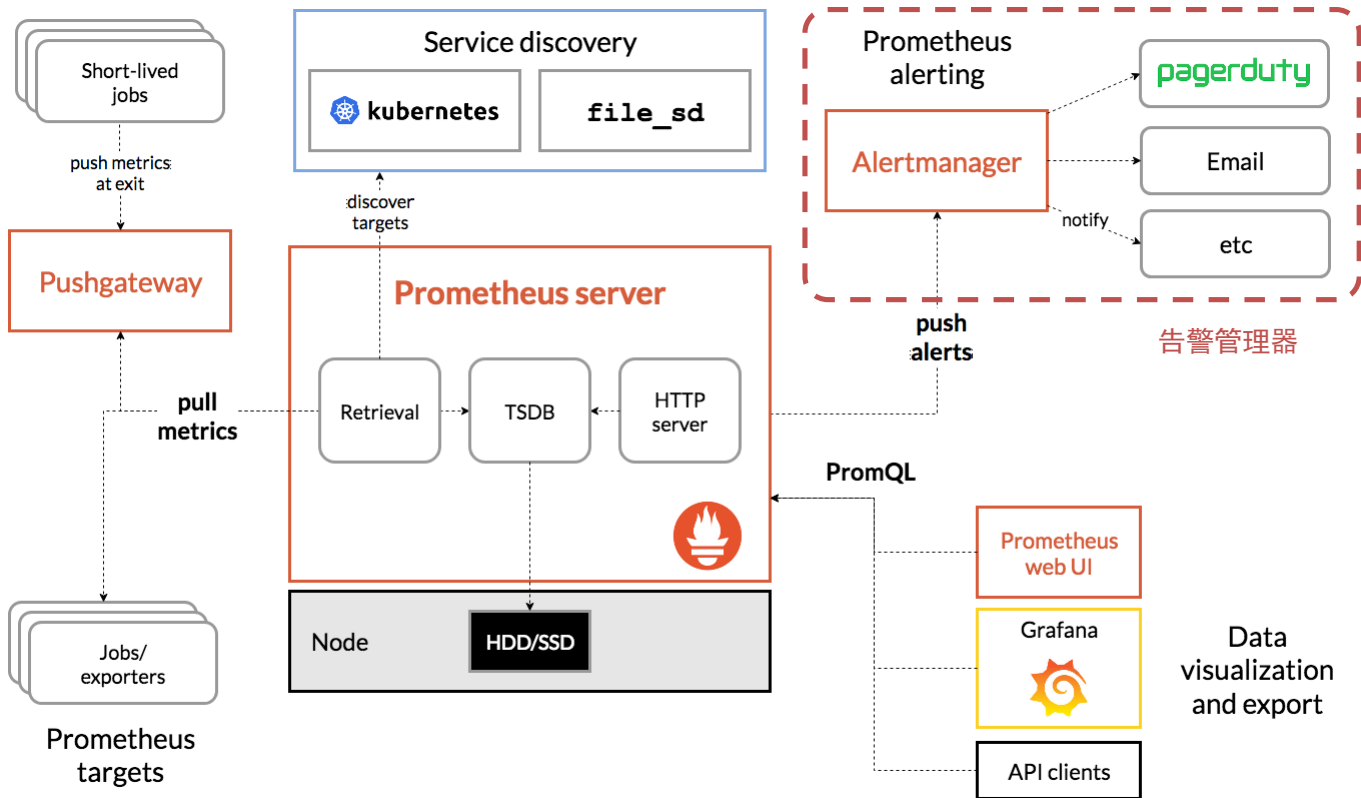
1. Prometheus 架构



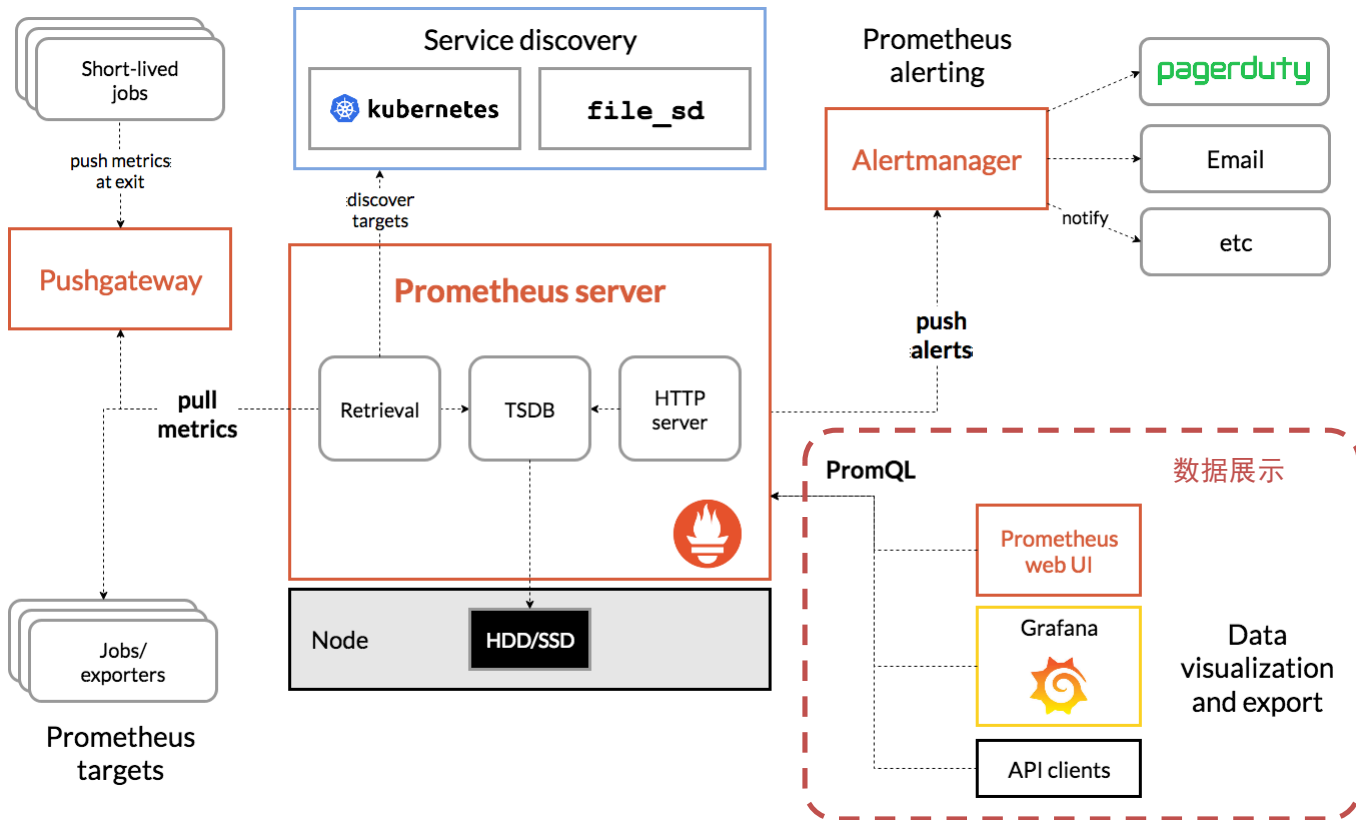
1. Prometheus 架构



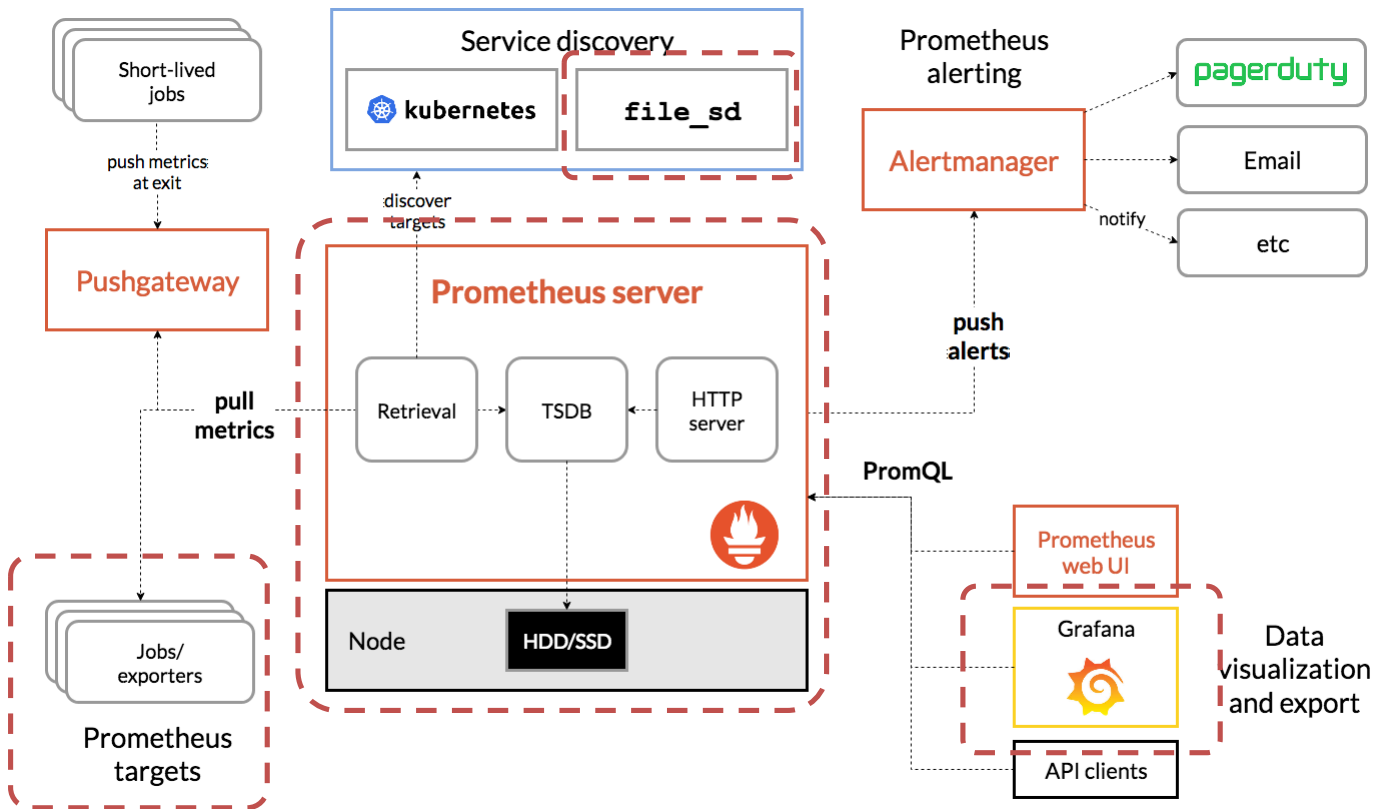
1. Prometheus 架构



1. Prometheus 架构



1. Prometheus 架构



■ 2. Prometheus + Grafana 监控实践

实践流程

SpringBoot 应用

1. 集成 micrometer

2. 监控 JVM 性能指标

8. 自定义监控指标

Prometheus

3. 部署

4. 对接 SpringBoot

9. 动态变更监控目标

Grafana

5. 部署

6. 对接 Prometheus

7. 添加 JVM 监控仪表盘



总结

重难点

1. Prometheus + Grafana 监控 SpringBoot 应用的方式

下节

实践 Springboot Admin



目录 Contents

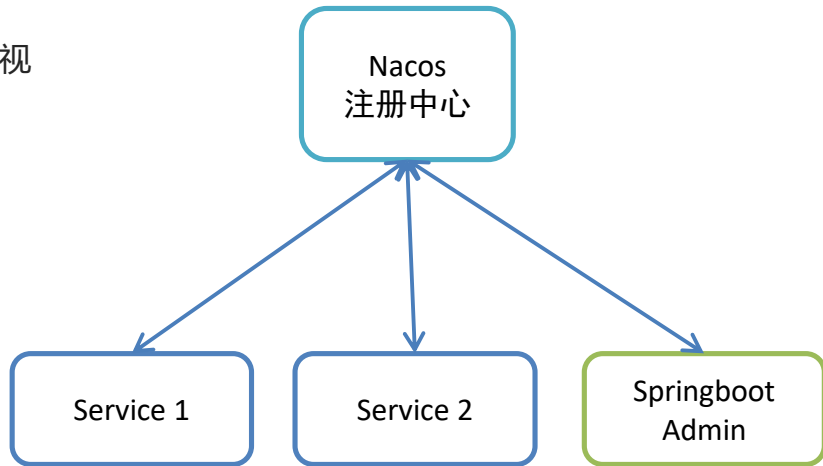
- ◆ Prometheus 监控实践
- ◆ Spring Boot Admin 监控实践
- ◆ JVM GC 日志可视化分析

小节导学

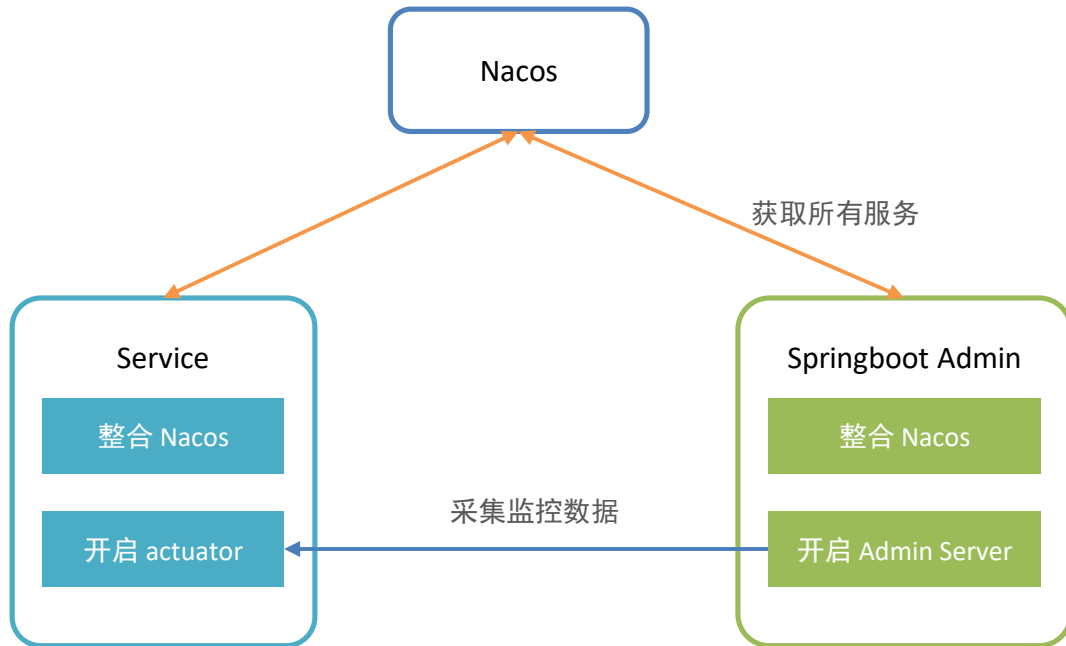
Springboot actuator 提供了丰富的监控数据，但这些数据并不适合我们查看，需要一个可视化工具。

Springboot admin 就是一款主流的 actuator 监控数据可视化工具。

业务服务与 Springboot Admin 都集成 Nacos 注册中心，Springboot Admin 即可自动监测所有服务。



整体结构



步骤 - 创建被监控服务

主要依赖:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
```


步骤 - 创建被监控服务

属性配置:

```
server:
  port: 8080
spring:
  application:
    name: service-provider
cloud:
  nacos:
    discovery:
      server-addr: localhost:8848
```

```
#开启端点
management:
  endpoints:
    web:
      exposure:
        include: '*'
  endpoint:
    health:
      show-details: ALWAYS
```

步骤 - 创建 Springboot Admin 监控服务

主要依赖:

```
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-starter-server</artifactId>
    <version>2.1.5</version>
</dependency>
```

步骤 - 创建 Springboot Admin 监控服务

属性配置:

```
server:
  port: 8081
spring:
  application:
    name: spring-boot-admin
  cloud:
    nacos:
      discovery:
        server-addr: localhost:8848
```

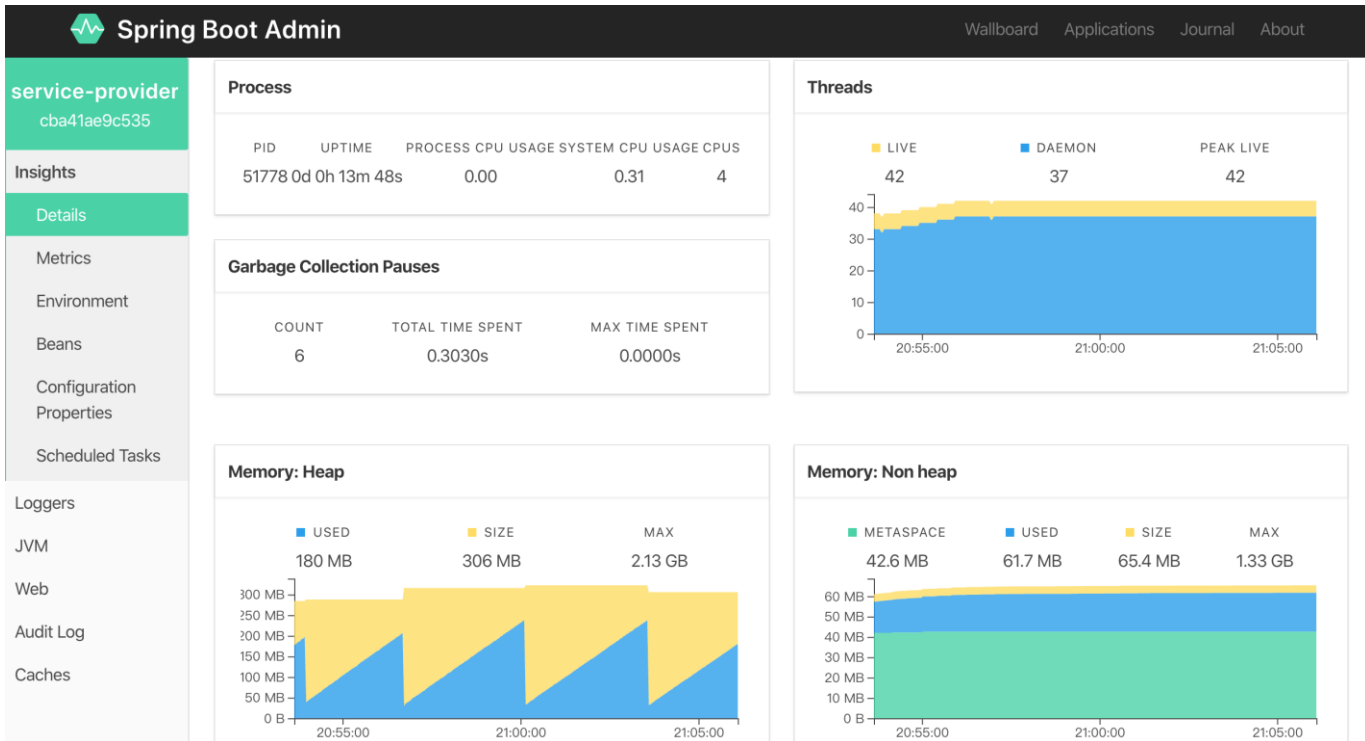
步骤 - 创建 Springboot Admin 监控服务

开启 Admin:

```
@EnableDiscoveryClient
@EnableAdminServer
@SpringBootApplication
public class SpringbootadminApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringbootadminApplication.class, args);
    }
}
```


Spring Boot Admin 监控实践

效果



Spring Boot Admin 监控实践

效果

 Spring Boot Admin

Wallboard Applications Journal About

service-provider
cba41ae9c535

Insights

Loggers

JVM

Web

Audit Log

Caches

502/502

☐ class only ☐ configured

ROOT	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba.cloud	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba.cloud.nacos	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba.cloud.nacos.NacosDiscoveryProperties	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba.cloud.nacos.discovery	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset
com.alibaba.cloud.nacos.discovery.NacosDiscoveryClient	OFF	ERROR	WARN	INFO	DEBUG	TRACE	Reset

Prometheus VS SpringbootAdmin

Prometheus

- 复杂
- 专业监控平台
- 可以存储时序大数据
- 支持告警
- 支持 K8S 等多种服务发现机制
- 扩展能力强

Springboot Admin

- 简单
- 功能单一
- 只适用于 Springboot 应用
- 没有存储
- 没有告警
- 扩展空间有限



总结

重难点

1. Spring Boot Admin 监控配置方法

下节

JVM 垃圾回收日志可视化分析



目录 Contents

- ◆ Prometheus 监控实践
- ◆ Spring Boot Admin 监控实践
- ◆ JVM GC 日志可视化分析

小节导学

JVM 的垃圾回收 GC 日志是我们的观察服务状态的重要信息。但 GC 日志的**可读性很差**，分析起来是极其痛苦的，所以也需要可视化分析工具。**gceasy.io** 是一个很方便的分析工具。



```
Java HotSpot(TM) 64-Bit Server VM (25.102-b14) for bsd-amd64 JRE (1.8.0_102-b14),  
built on Jun 22 2016 11:42:36 by "java_re" with gcc 4.2.1 (Based on Apple Inc. build  
5658) (LLVM build 2336.11.00)  
Memory: 4k page, physical 8388608k(234176k free)
```

/proc/meminfo:

```
CommandLine flags: -XX:-BytecodeVerificationLocal -XX:-BytecodeVerificationRemote  
-XX:InitialHeapSize=5242880 -XX:+ManagementServer -XX:MaxHeapSize=2147483648 -XX:  
+PrintGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:TieredStopAtLevel=1 -XX:  
+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseParallelGC  
0.330: [GC (Allocation Failure) [PSYoungGen: 1011K->512K(1536K)] 1011K->520K(5632K),  
0.0110647 secs] [Times: user=0.00 sys=0.00, real=0.02 secs]  
0.472: [GC (Allocation Failure) [PSYoungGen: 1529K->512K(2560K)] 1537K->699K(6656K),  
0.0009812 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  
0.554: [GC (Allocation Failure) [PSYoungGen: 2560K->512K(2560K)] 2747K->894K(6656K),  
0.0011020 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  
0.663: [GC (Allocation Failure) [PSYoungGen: 2560K->503K(4608K)] 2942K->1222K(8704K),  
0.0015554 secs] [Times: user=0.01 sys=0.00, real=0.00 secs]  
0.788: [GC (Allocation Failure) [PSYoungGen: 4599K->503K(4608K)] 5318K->1882K(8704K),  
0.0018599 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]
```

步骤 - 产生 GC 日志

VM 参数配置:

```
-XX:+PrintGCDetails -Xloggc:gc.log
```

Run/Debug Configurations

Name: ServiceproviderApplication

Configuration

Code Coverage

Logs

Main class: com.example.demo.ServiceproviderApplication

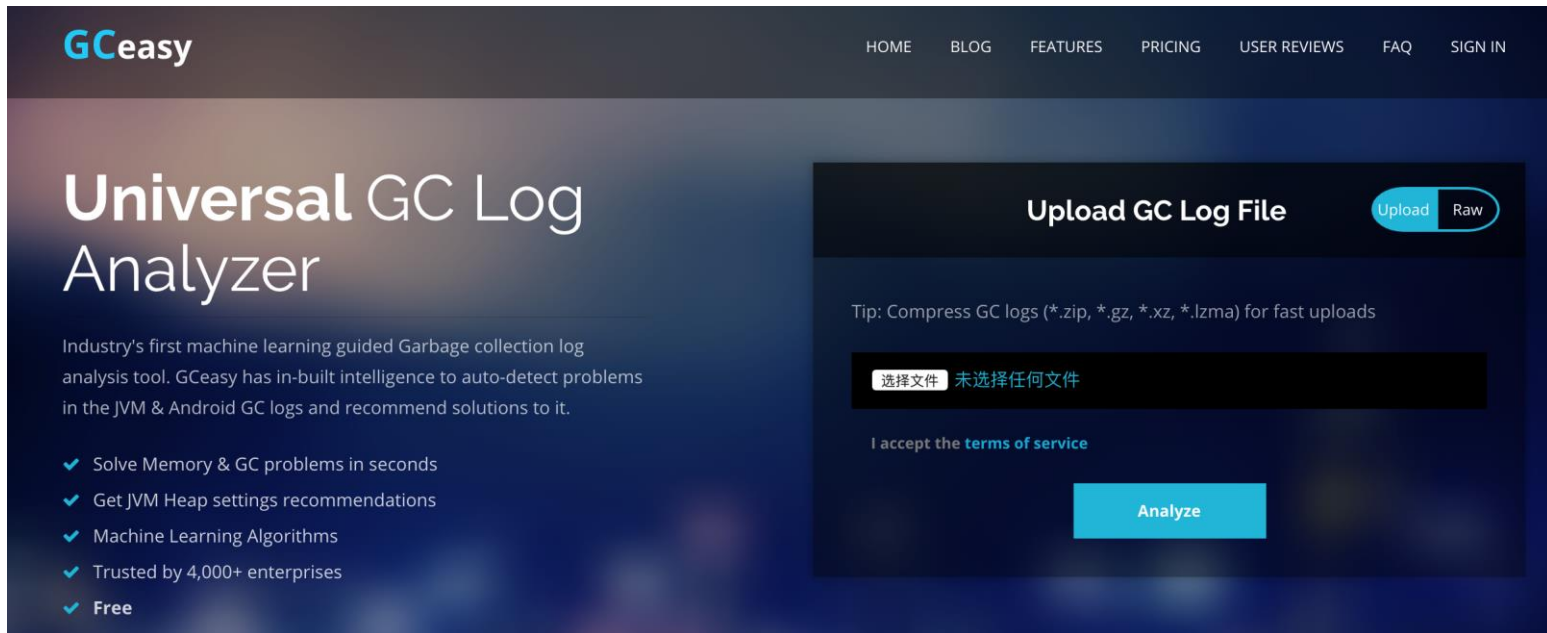
▼ Environment

VM options: -XX:+PrintGCDetails -Xloggc:gc.log

Program arguments:

步骤 - 分析 GC 日志

访问 <https://gceasy.io/>



步骤 - 分析 GC 日志

访问 <https://gceasy.io/>

GCeasy

HOME

BLOG

FEATURES

PRICING

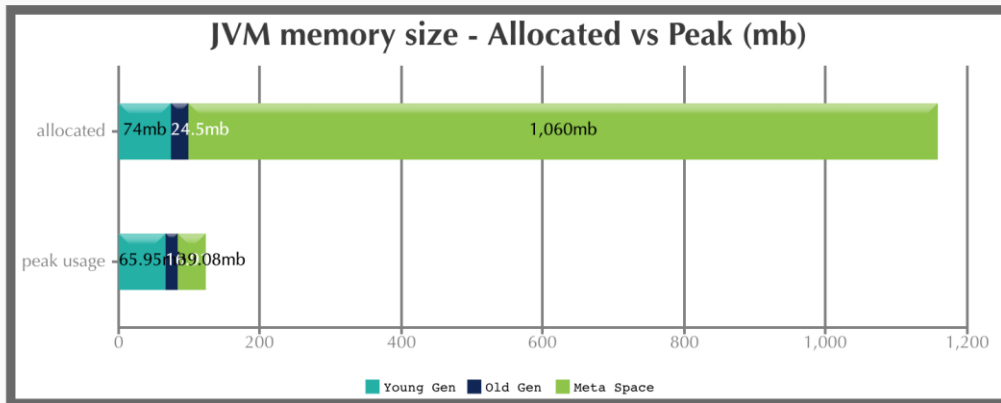
USER REVIEWS

FAQ

SIGN IN

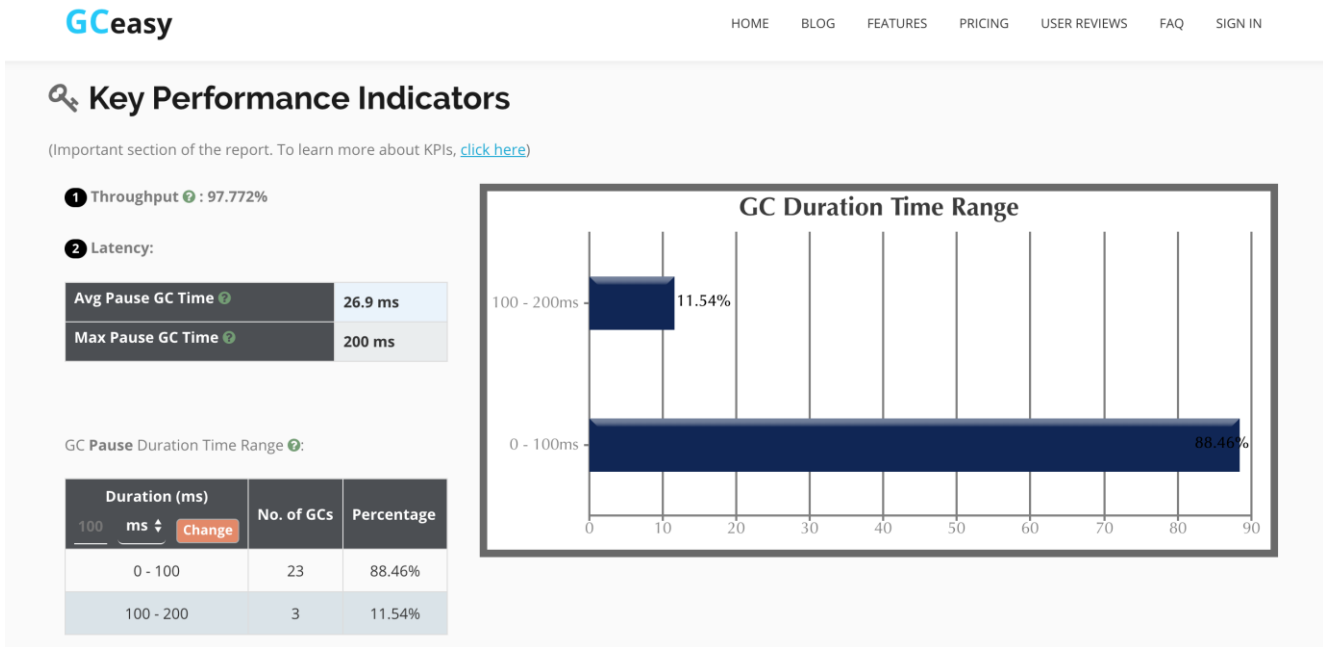
JVM memory size

Generation	Allocated	Peak
Young Generation	74 mb	65.95 mb
Old Generation	24.5 mb	16.91 mb
Meta Space	1.04 gb	39.08 mb
Young + Old + Meta space	3.04 gb	118.11 mb



步骤 - 分析 GC 日志

访问 <https://gceasy.io/>





总结

重难点

1. GC 日志产生方法
2. gceasy.io 分析 GC 日志的方法



总结

重难点

1. GC 日志产生方法
2. gceasy.io 分析 GC 日志的方法

下节

本章内容总结



一样的在线教育，不一样的教学品质