

KNode

— 打造快手高性能Node.js应用

张岫

平台研发部工程师



自我介绍



张岫

个人经历:

2011 ~ 2015 搜狗

2015 ~ 2018 百度

2018 ~ 至今 快手

目录

1. KNode是什么？产生的背景？
2. Node.js基础监控能力建设
3. KNode展望未来

KNode是什么

狭义上：KNode是一个定制的Node.js运行时

广义上：预期做成快手一站式的Node.js解决方案

KNode产生的背景

1. 快手各业务线都有Node.js的身影

KNode产生的背景

BFF 点播 *express* 主站 音视频
eggjs
海外 *BigPipe* 短视频 商业化
直播 *koa* 电商
GraphQL *AjaxPipe*
中台

KNode产生的背景

1. 快手各业务线都有Node.js的身影
2. 春晚一战暴露了一些问题
 - 基础监控粒度较粗，排障能力不足
 - 各业务团队方案不一，有简有繁
3. 技术团队想在Node.js领域深耕

Node.js基础监控能力建设

如何做好监控、提升排障能力？



Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

- 常规日志：服务启动后间隔输出运行情况
- 按需日志：服务出现不符预期的时候触发

Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

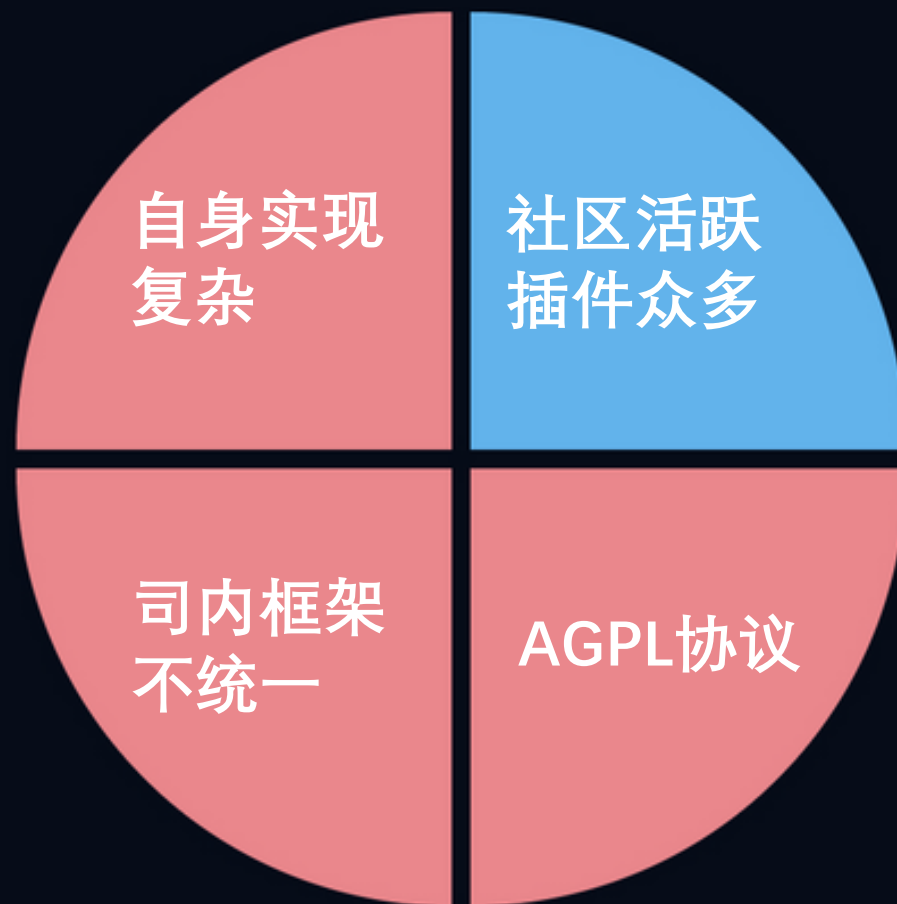
- 通过pm2的监控插件和日志
- 通过Node.js扩展或js模块的方式
- 通过定制Node.js源代码

Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

- 通过pm2的监控插件和日志
- 通过Node.js扩展或js模块的方式
- 通过定制Node.js源代码

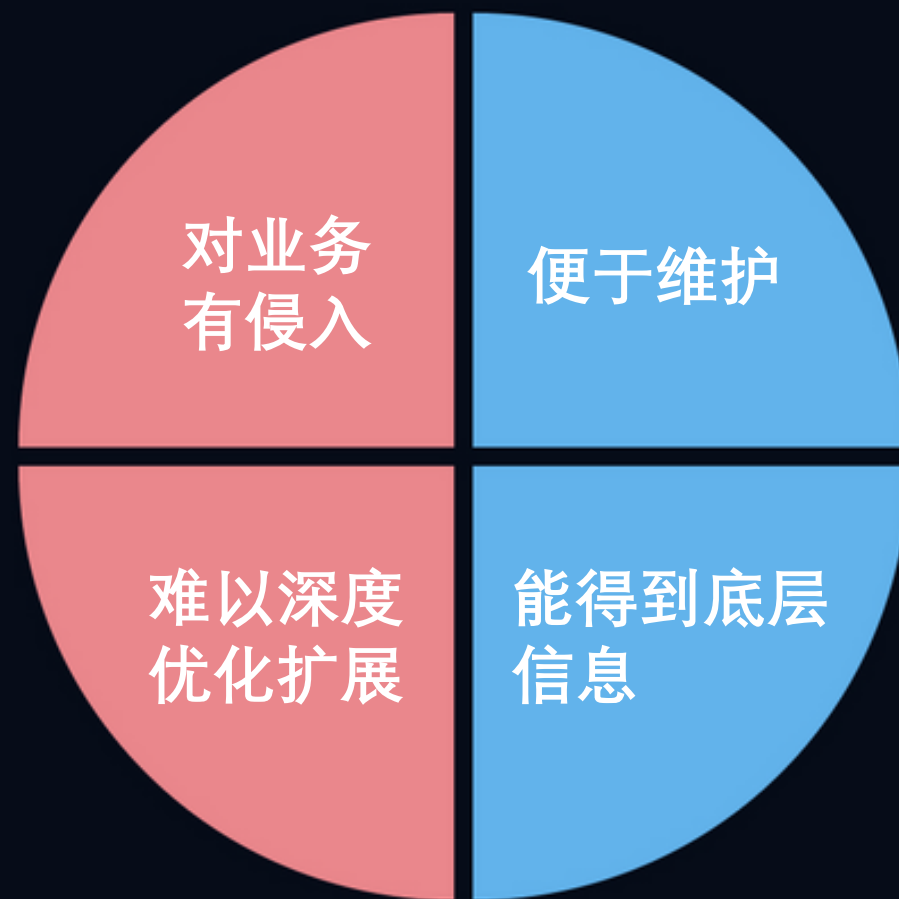


Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

- 通过pm2的监控插件和日志
- 通过Node.js扩展或js模块的方式
- 通过定制Node.js源代码

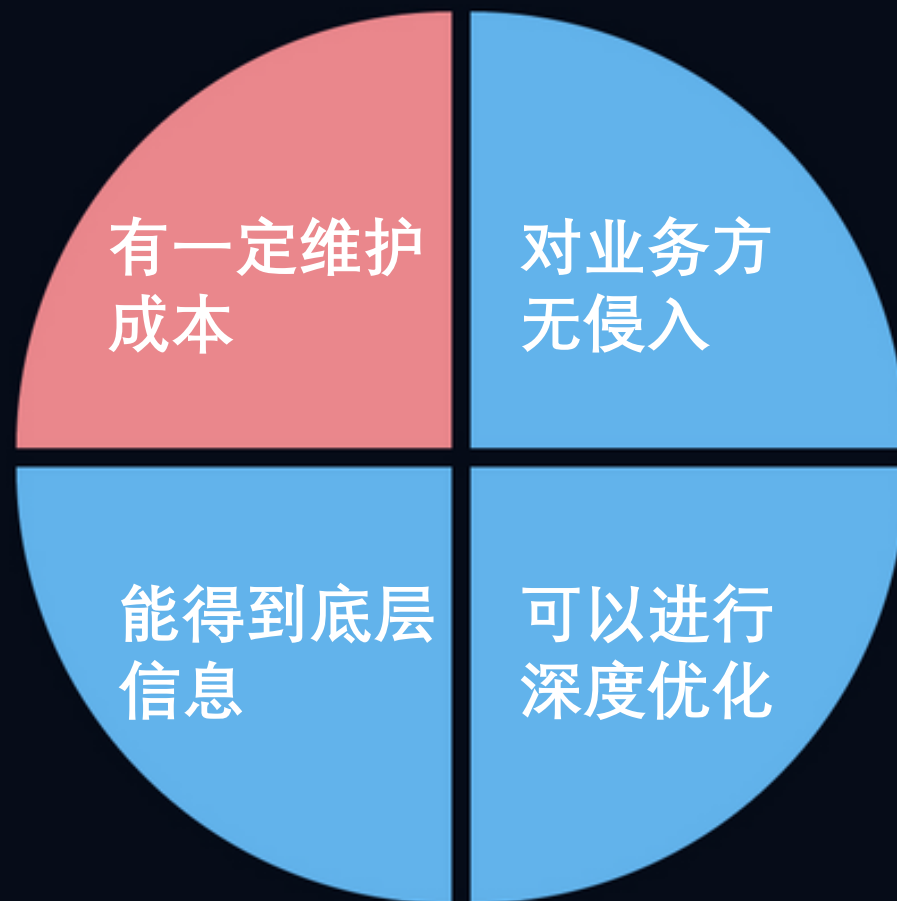


Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

- 通过pm2的监控插件和日志
- 通过Node.js扩展或js模块的方式
- 通过定制Node.js源代码



Node.js基础监控能力建设

如何做好监控、提升排障能力？

数据生产

定制运行时

数据采集

容器环境统一日志收集器

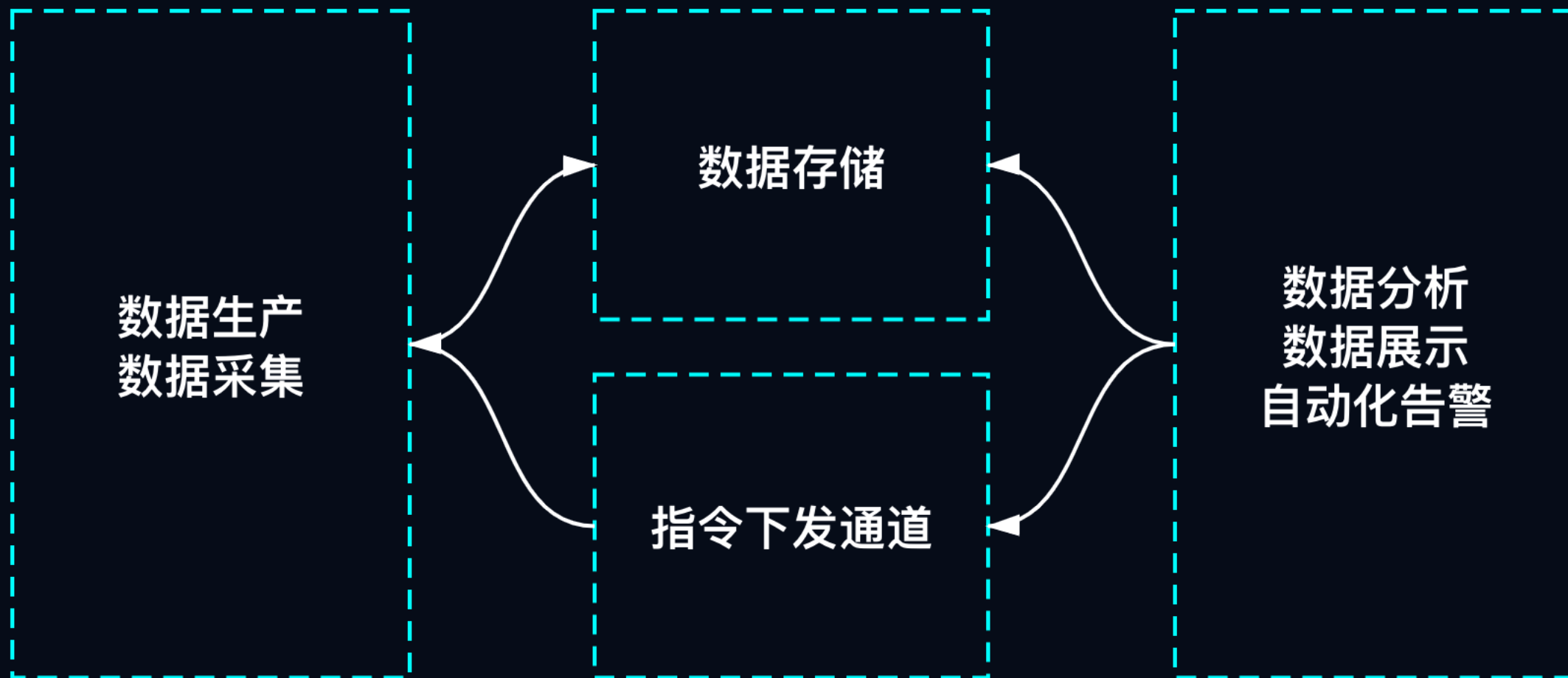
数据展现
与分析

提供平台，集成dashboard和devtool

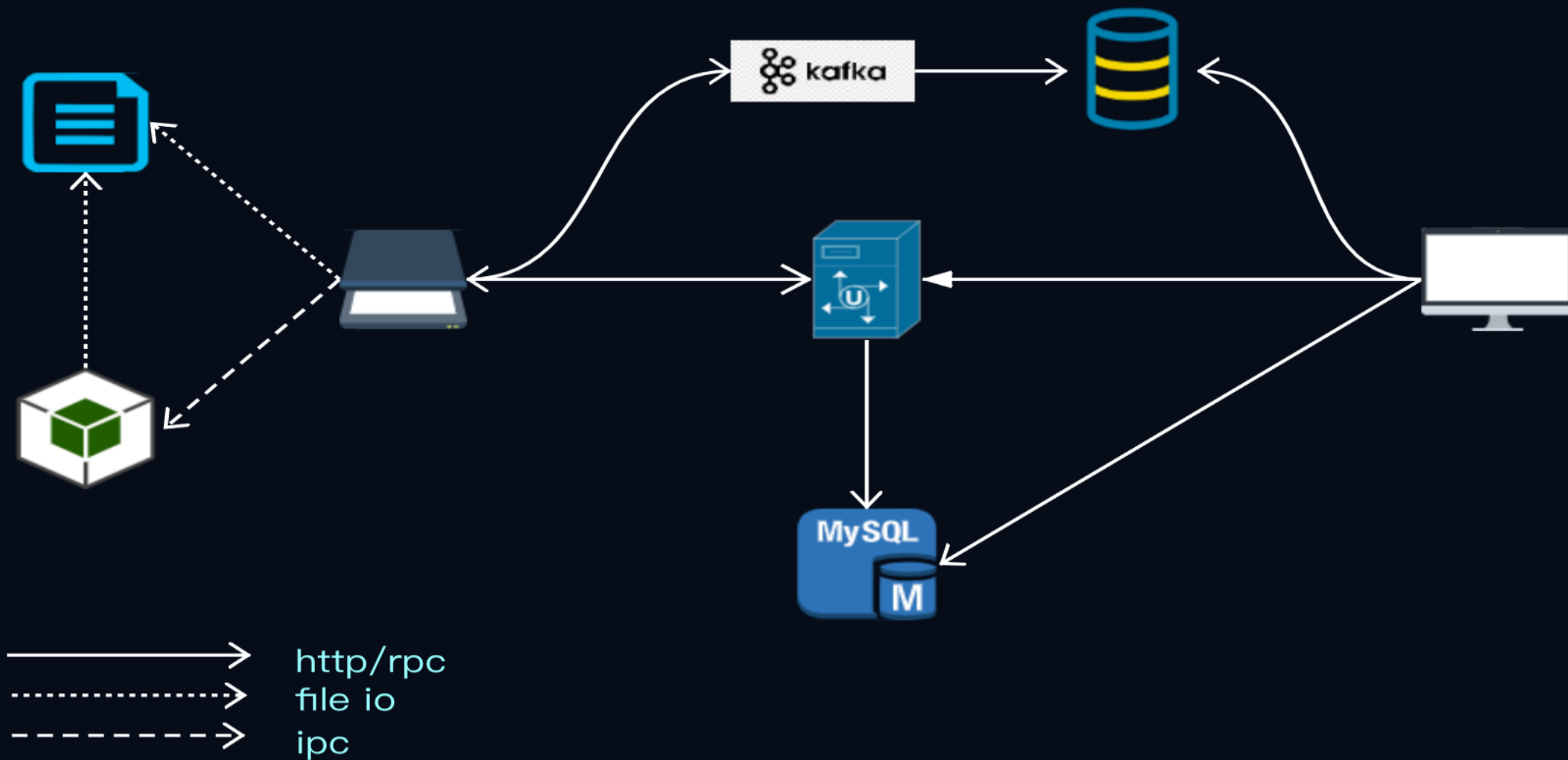
自动化告
警

平台打通司内基于prometheus的告警系统

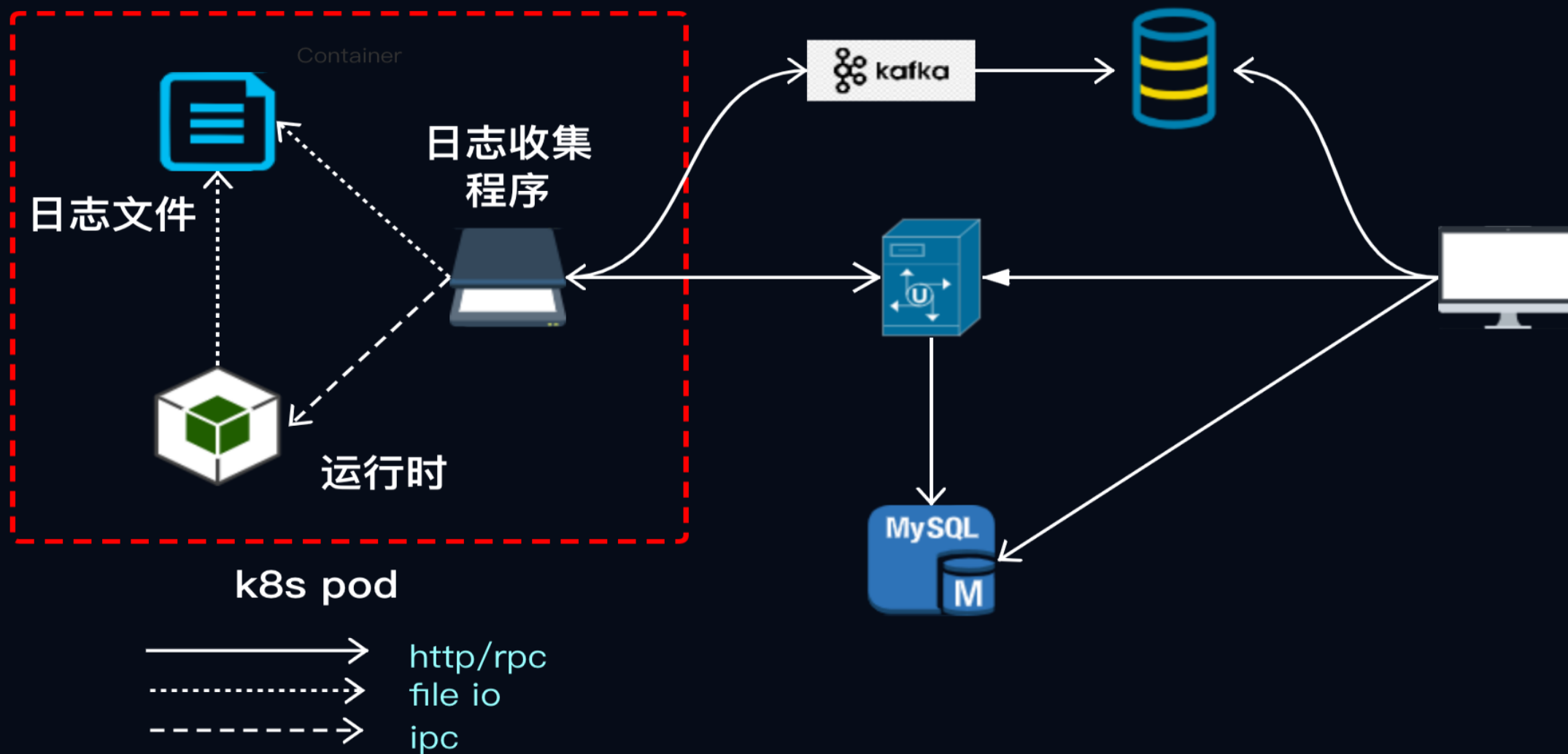
Node.js基础监控能力建设



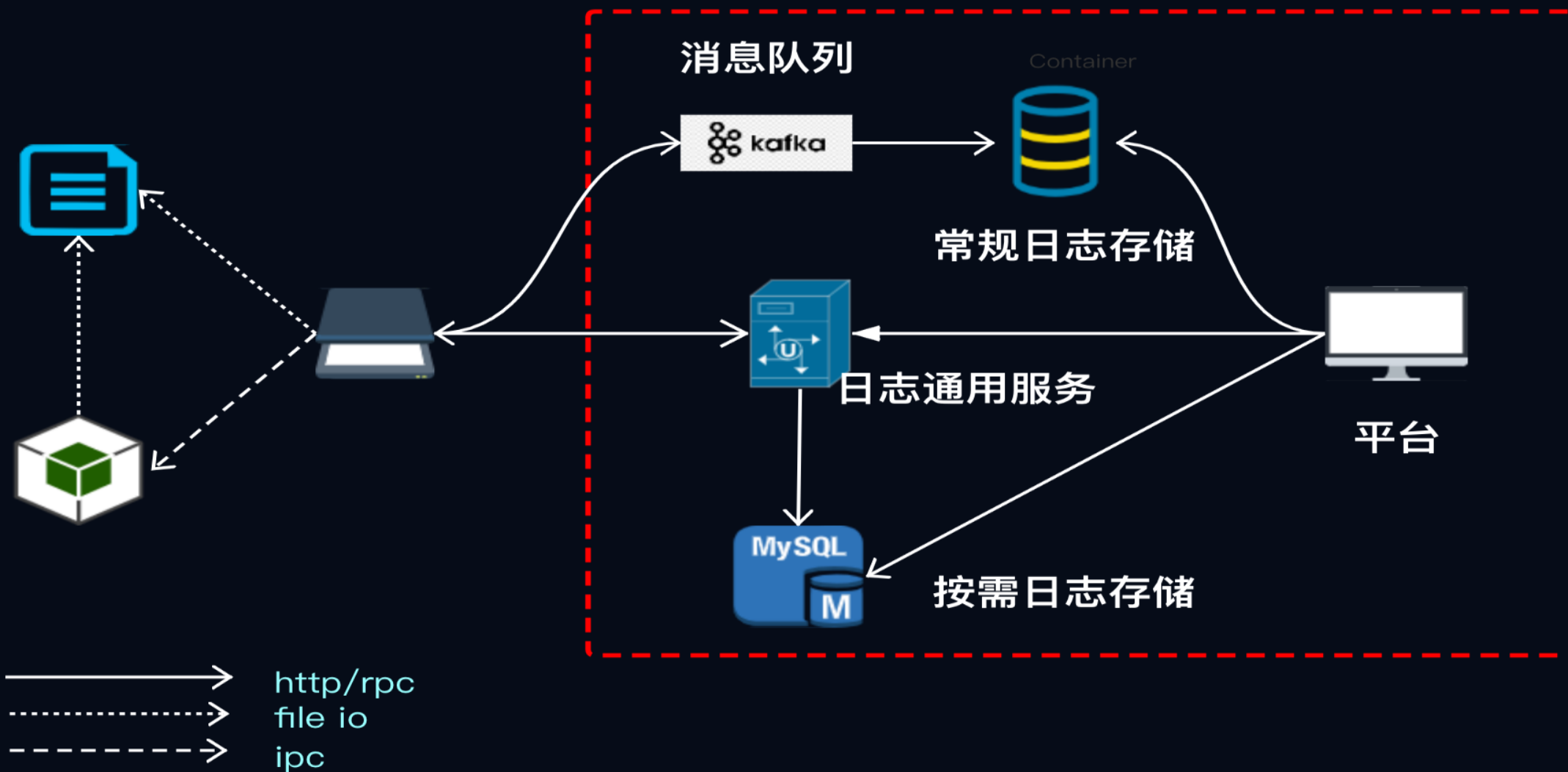
Node.js基础监控能力建设



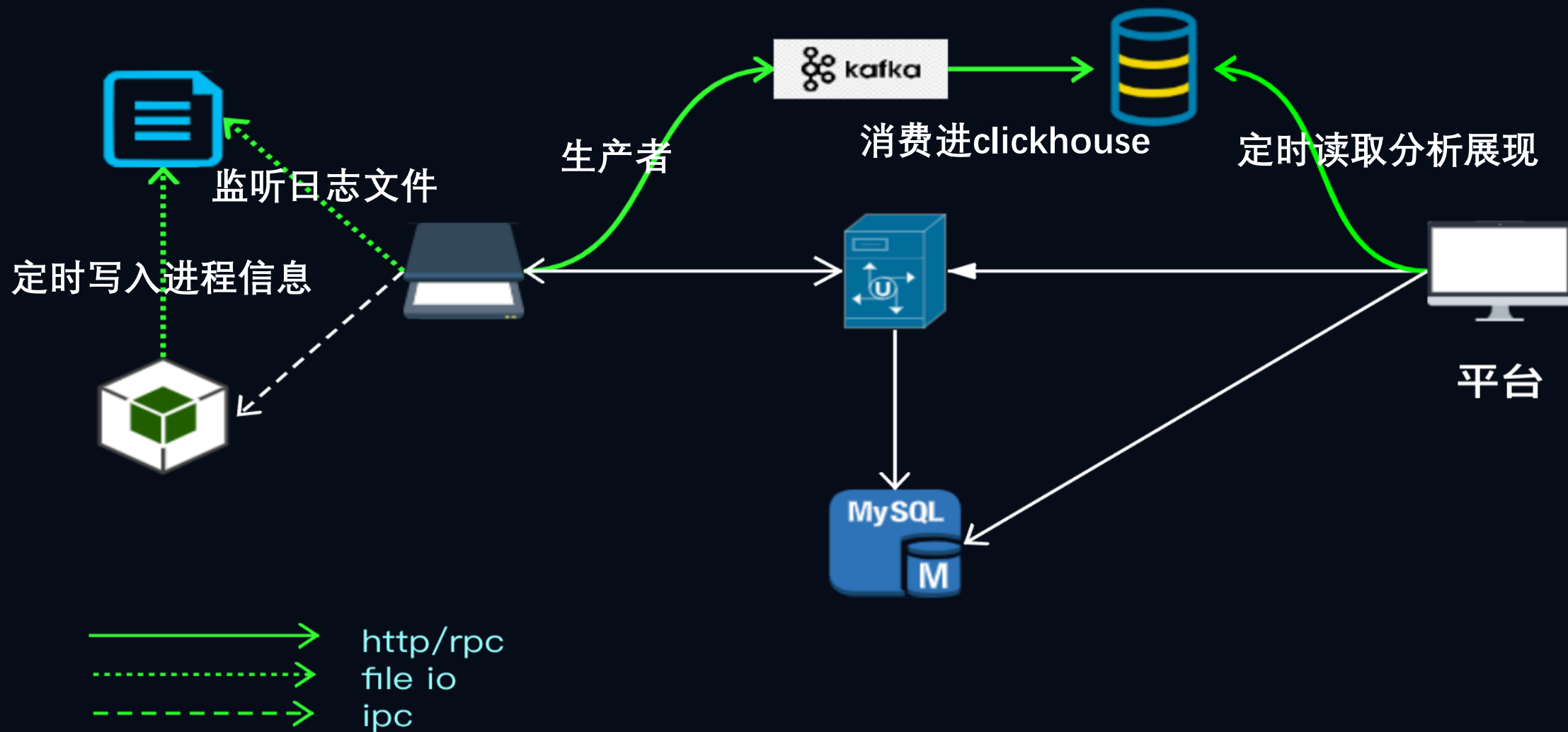
Node.js基础监控能力建设 – 前端



Node.js基础监控能力建设 - 后端



Node.js基础监控能力建设 – 常规日志



Node.js基础监控能力建设 – 常规日志

heap:

JavaScript

```
const v8 = require('v8');  
v8.getHeapStatistics();
```

C++

```
v8::HeapStatistics v8_heap_stats;  
isolate->GetHeapStatistics(&v8_heap_stats);
```

Node.js基础监控能力建设 – 常规日志

heap space:

JavaScript

```
const v8 = require('v8');  
v8.getHeapSpaceStatistics();
```

C++

```
v8::HeapSpaceStatistics v8_heap_spaces;  
isolate->GetHeapSpaceStatistics(&v8_heap_spaces, index);
```

Node.js基础监控能力建设 – 常规日志

cpu:

JavaScript

```
process.cpuUsage(prevValue);
```

C++

```
uv_rusage_t rusage;  
int result = uv_getrusage(&rusage);
```

```
now_cpu_usage = clock();  
now_time = uv_hrtime();
```

Node.js基础监控能力建设 – 常规日志

gc:

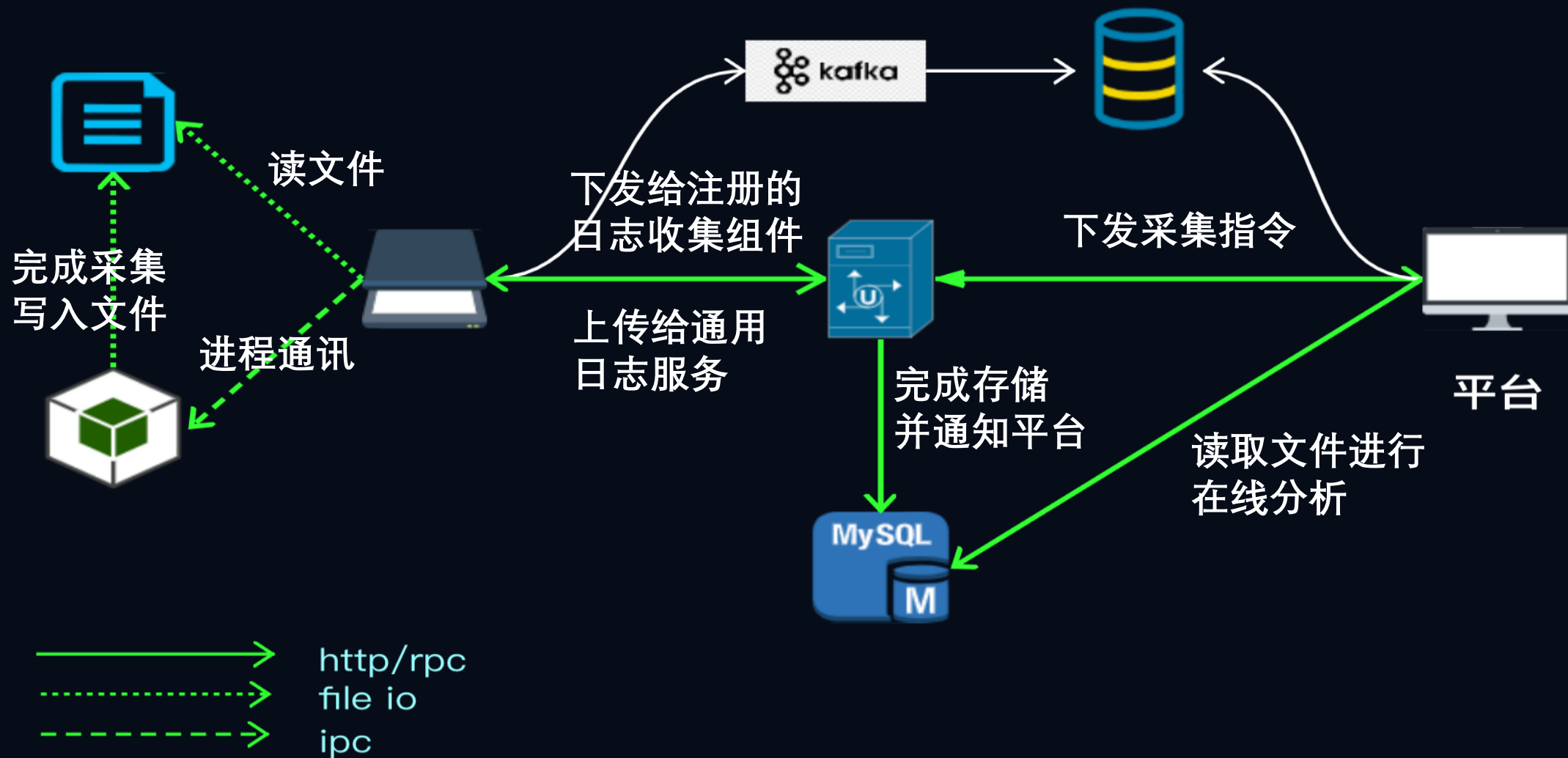
C++

```
void GCPilogueCallback(  
    v8::Isolate* isolate,  
    v8::GCType type,  
    v8::GCCallbackFlags flags);
```

```
void GCEpilogueCallback(  
    v8::Isolate* isolate,  
    v8::GCType type,  
    v8::GCCallbackFlags flags);
```

```
isolate->AddGCPilogueCallback(GCPilogueCallback);  
isolate->AddGCEpilogueCallback(GCEpilogueCallback);
```


Node.js基础监控能力建设 – 按需日志



Node.js基础监控能力建设 – 按需日志

heapdump:

C++

```
v8::HeapSnapshot* snapshot = isolate->GetHeapProfiler()->TakeHeapSnapshot();  
ks::helper::FileOutputStream stream(fp);  
snapshot->Serialize(&stream, v8::HeapSnapshot::kJSON);
```

Node.js基础监控能力建设 – 按需日志

cpu profile:

C++

```
cpu_profiler = new ks::CPUProfiler();  
cpu_profiler->StartProfiling(title, true);  
std::string result = cpu_profiler->StopProfiling(title);
```

Node.js基础监控能力建设 – 按需日志

others:

C++

```
heap_profiler->StartSamplingHeapProfiler();  
v8::AllocationProfile* profile = heap_profiler->GetAllocationProfile();  
heap_profiler->StopSamplingHeapProfiler();
```

```
heap_profiler->StartTrackingHeapObjects(true);  
heap_profiler->StopTrackingHeapObjects();
```

```
isolate->LowMemoryNotification();
```

```
isolate->GetHeapCodeAndMetadataStatistics();
```

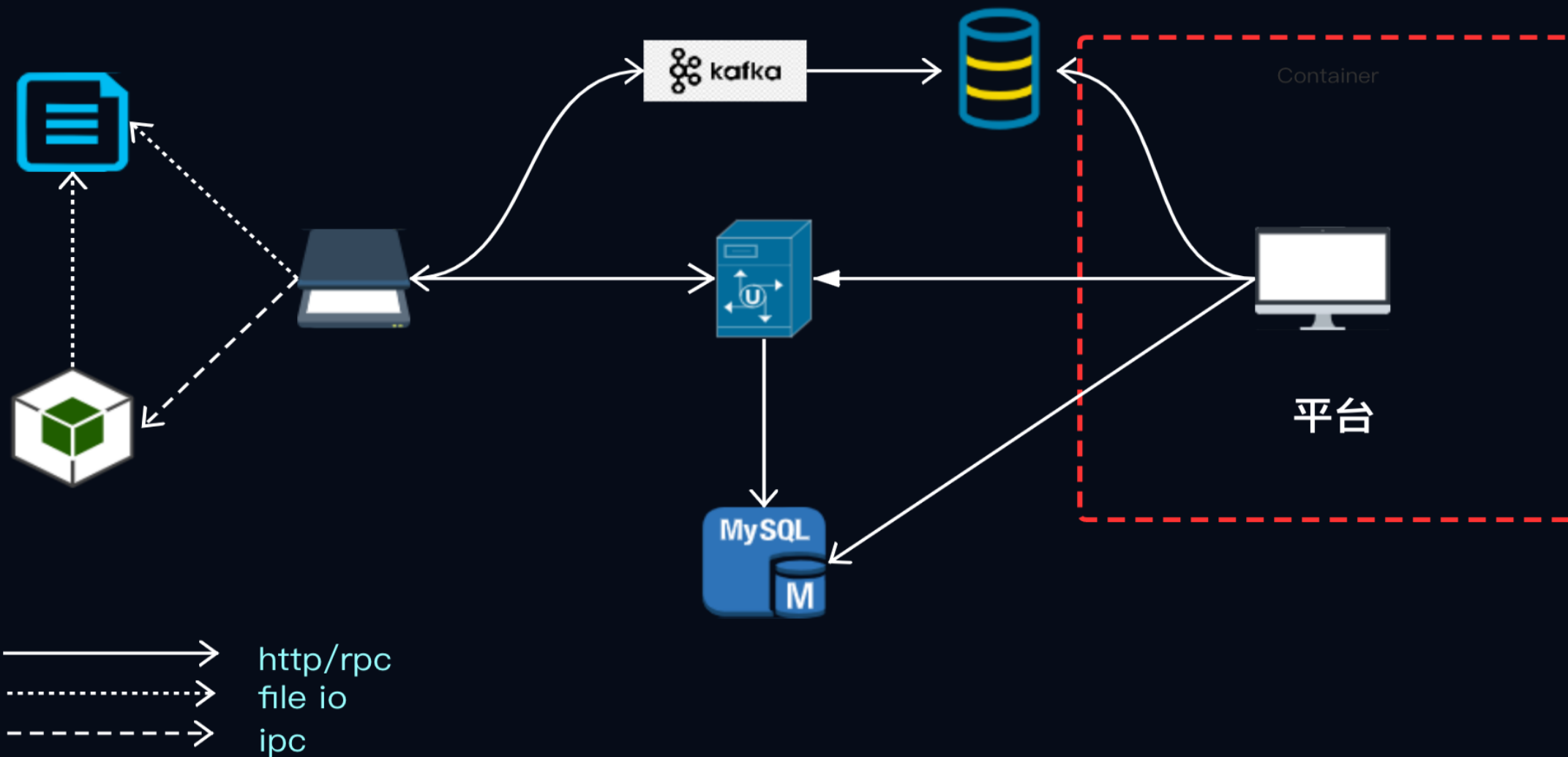
```
isolate->GetHeapObjectStatisticsAtLastGC()
```

Node.js基础监控能力建设

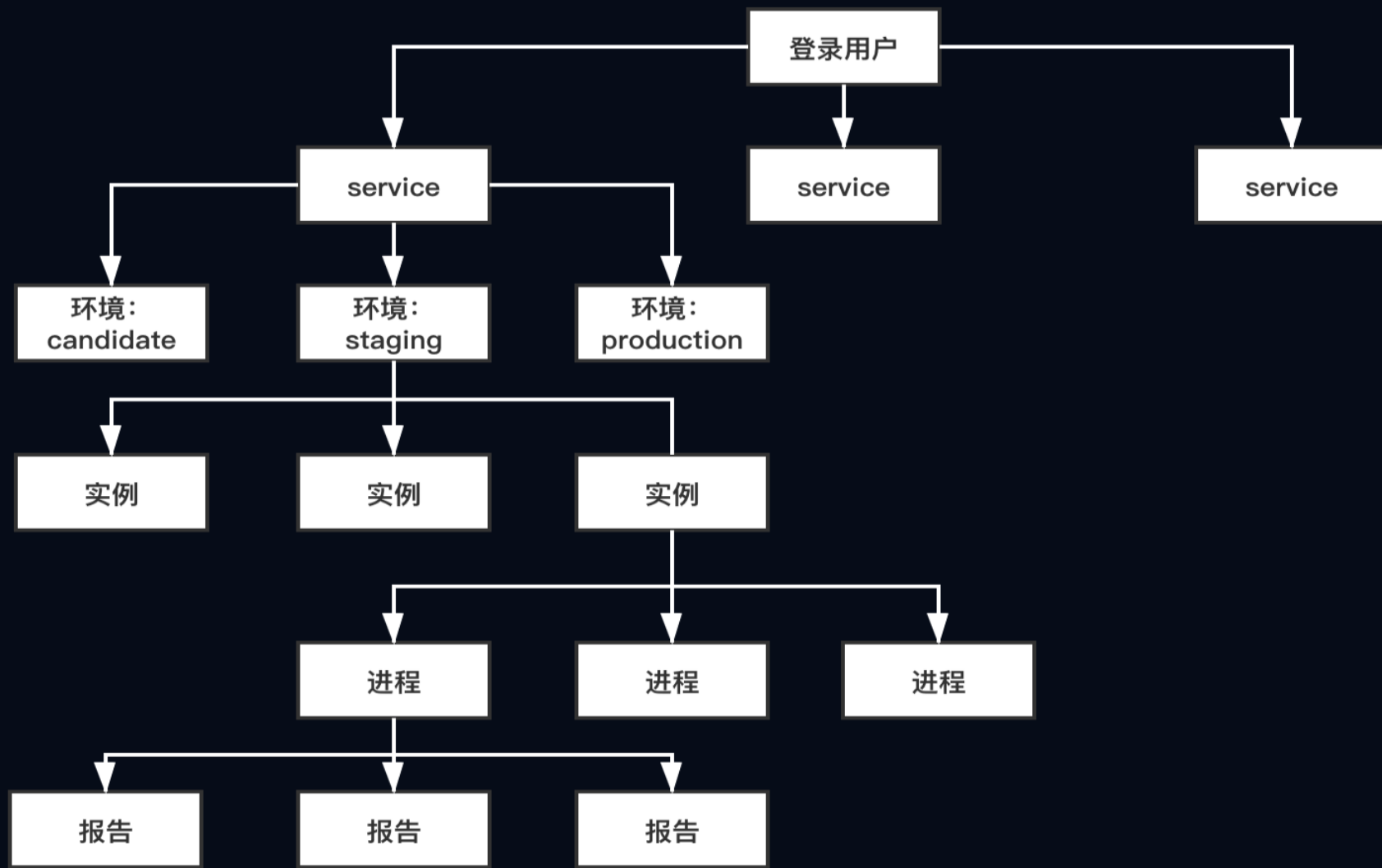
注意事项:

1. 常规日志系统写方法、日志切割
2. 容器环境IPC的限制
3. 多线程处理
4. 其他容错处理

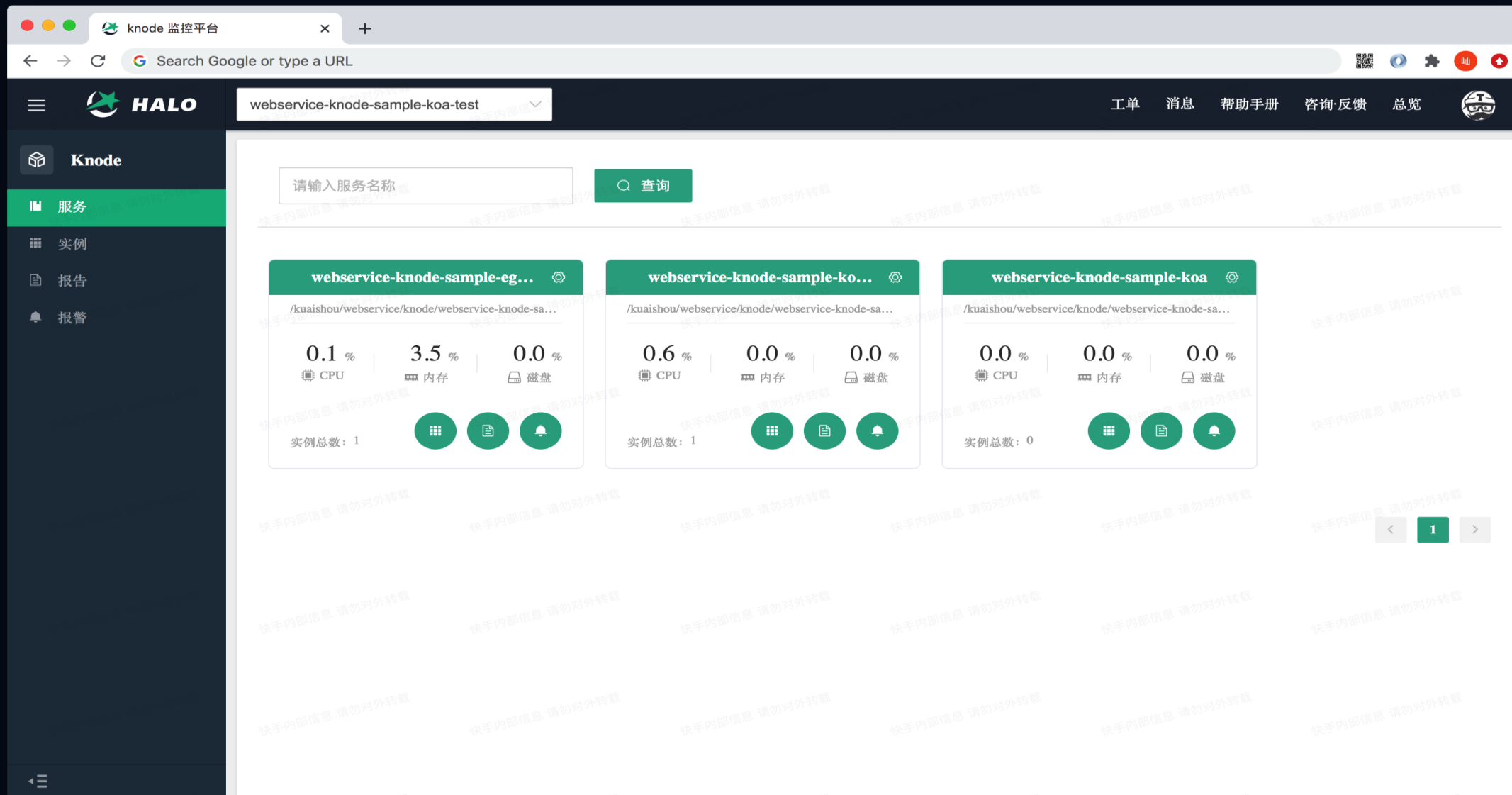
Node.js基础监控能力建设 - 平台



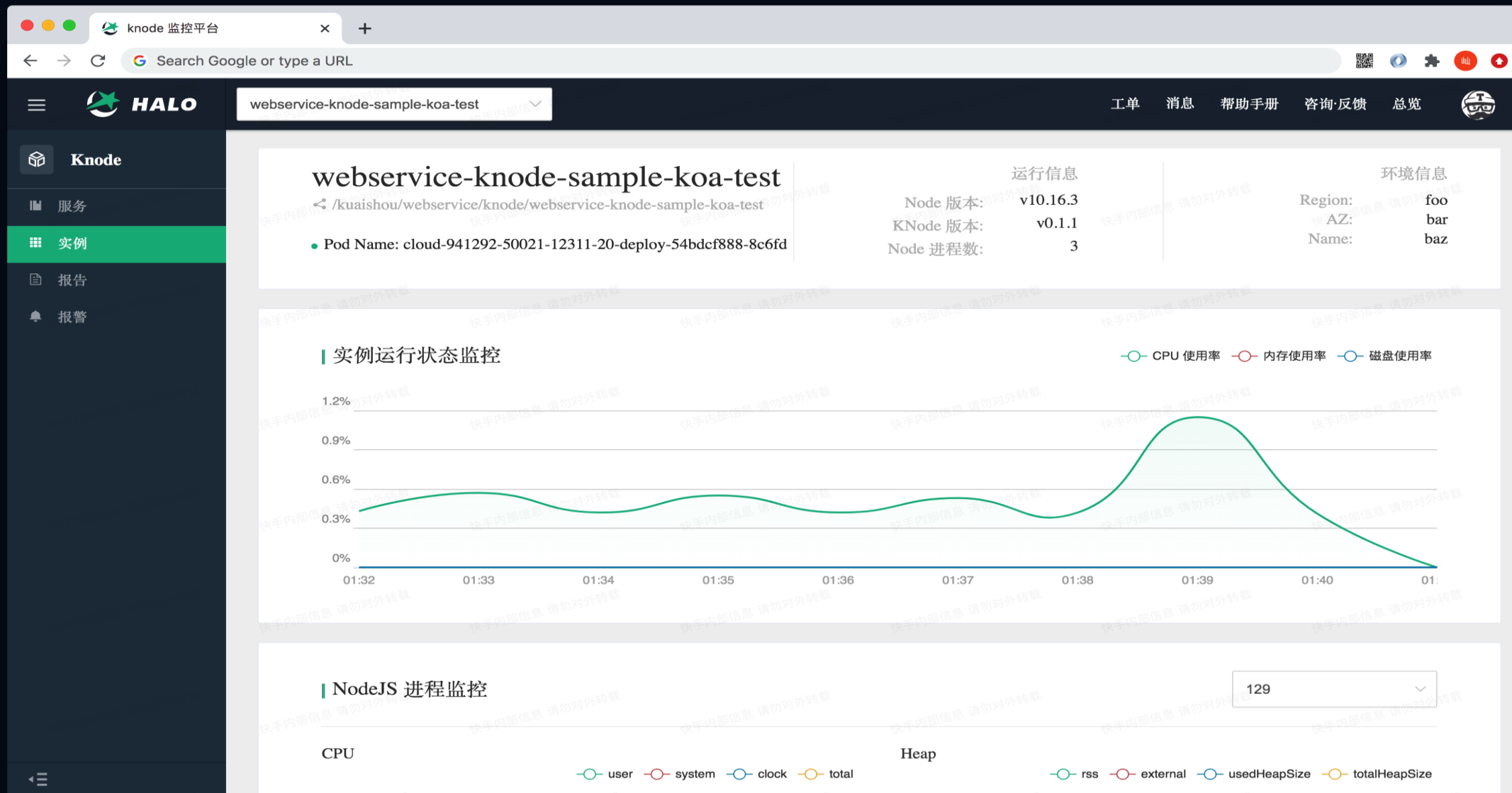
Node.js基础监控能力建设 - 平台



Node.js基础监控能力建设 - 平台



Node.js基础监控能力建设 - 平台



Node.js基础监控能力建设 - 平台

knode 监控平台

Search Google or type a URL

咨询·反馈 总览

HALO

Knode

服务

实例

报告

报警

报警规则设置

规则配置

cpu.usage.percent

avg

>=

70

报警配置

报警名 (最多50字符)

☒ 是否接收恢复通知

☒ 是否接收重复提醒

报警消息 (最多50字符)

报警规则运行间隔

保持多久后会报警

最大报警次数

环境作用域

接收报警

报警接收人

报警等级

KIM token

☐ 手机

☐ 短信

☐ KIM

☒ 邮件

其他

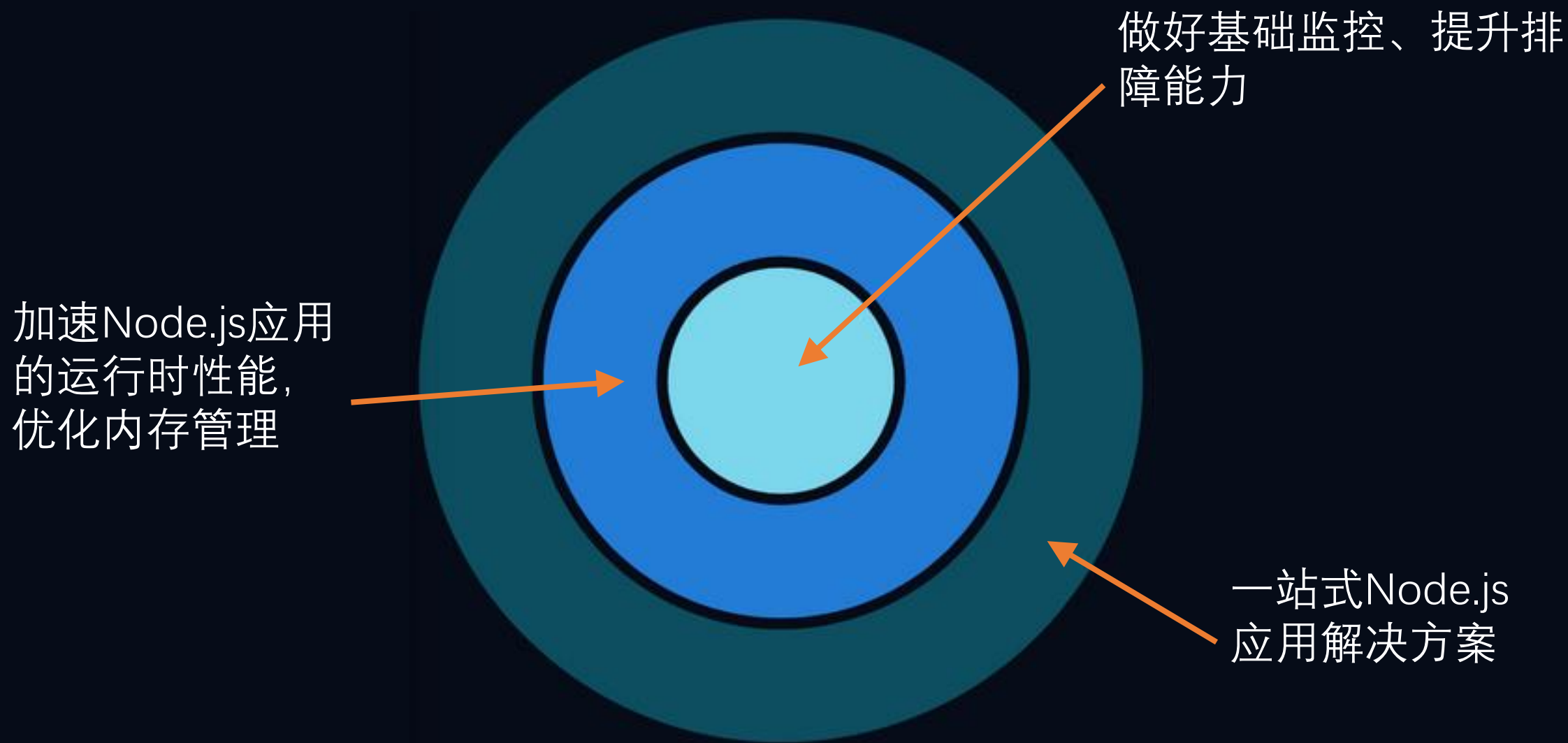
报警描述

该信息不会显示在报警消息中 (最多100字符)

Node.js基础监控能力建设 – 平台

Summary		Class filter		All objects					
Profiles		HEAP SNAPSHOTS		dmo-acfun-frontend-h5-kcs		4.5 MB			
Constructor		Distance		Shallow Size		Retained Size			
▶ (compiled code) x3575		3		817 392 18 %		2 527 632 57 %			
▼ (closure) x4282		2		257 712 6 %		1 631 236 37 %			
▶ NativeModule() @33599		6		64 0 %		373 512 8 %			
▶ ClientRequest() @68307		10		64 0 %		37 160 1 %			
▶ Module() @49453 □		5		64 0 %		27 504 1 %			
▶ split() @60995 □		4		56 0 %		22 168 0 %			
▶ Object() @38827 □		2		64 0 %		19 664 0 %			
▶ startup() @34435		6		64 0 %		15 728 0 %			
▶ AssertionError() @34381		5		64 0 %		13 840 0 %			
▶ test() @25573 □		4		56 0 %		12 176 0 %			
▶ filter() @59761 □		4		56 0 %		12 168 0 %			
▶ slice() @49435 □		4		56 0 %		11 304 0 %			
▶ assign() @59485 □		3		56 0 %		8 864 0 %			
▶ Socket() @72103		8		64 0 %		8 856 0 %			
▶ URL() @49649 □		2		64 0 %		7 608 0 %			
▶ getOwnPropertyDescriptor() @48971 □		3		56 0 %		7 456 0 %			
▶ Map() @47685 □		2		64 0 %		6 296 0 %			
▶ RoundRobinHandle() @72353		8		64 0 %		6 224 0 %			
▶ internalBinding() @31813		5		64 0 %		6 120 0 %			
▶ RegExp() @59879 □		2		64 0 %		6 008 0 %			
▶ WeakMap() @61311 □		2		64 0 %		5 960 0 %			
▶ Buffer() @20937 □		2		64 0 %		5 624 0 %			
▶ realpathSync() @12869		8		64 0 %		5 392 0 %			
▶ pipeline() @14907 □		6		64 0 %		5 256 0 %			
▶ eos() @14915 □		6		64 0 %		4 944 0 %			
▶ () @30121		10		64 0 %		4 888 0 %			
▶ _uint8ArrayToBuffer() @26515 □		6		64 0 %		4 736 0 %			
▶ Date() @27291 □		2		64 0 %		4 656 0 %			
▶ forEach() @28271 □		3		56 0 %		4 592 0 %			
▶ Worker() @72409		6		64 0 %		4 552 0 %			
▶ Date() @60297 □		2		64 0 %		4 544 0 %			
▶ () @49277 □		6		64 0 %		4 424 0 %			
▶ getColorDepth() @59009 □		5		64 0 %		4 376 0 %			
▶ encodeStr() @53393		4		64 0 %		4 344 0 %			
▶ Set() @61101 □		2		64 0 %		4 176 0 %			
▶ next() @28973		4		56 0 %		3 984 0 %			
Retainers		Distance		Shallow Size		Retained Size			
Object									

展望未来



快手大前端
技术交流会 2020

THANKS

