

Takin-SRE -产品使用手册

使用过程中遇到产品使用的问题、功能 bug、产品使用体验、新的需求或意见，都可以提出来，我们一起让 Takin-SRE 完善起来~

快速了解 Takin-SRE

Takin-SRE 是什么？

Takin-SRE，在微服务架构下可以精准快速的确认业务健康状况和问题问题。使用 Takin-SRE 内置的数据和定位经验，问题确认和定位的时间可以从小时级别降低到 5 分钟以内。

名词解释

关键用户旅程，简称 CUJ	用户为实现一个目标与服务进行的一组互动，例如单次点击或多步骤流水线。 示例：“用户点击‘结算’按钮，等待购物车处理完成，系统返回收据。”
TestCase，简称 TC	作用在每个微服务接口、每个中间件请求上的保障用例，例如：是否异常的用例、是否超时的用例、业务是否失败的用例
不可用事件	每次请求，若直接影响到用户（包括响应异常、响应失败、响应严重超时），都将计为一次错误事件
不可用用户	每次请求，若直接影响到用户（包括响应异常、响应失败、响应严重超时），都将计为一次错误事件，该用户则计为一个不可用用户

产品特性

V0.1

业务还原能力：将核心用户旅程的微服务架构集中还原到一页纸上，告别系统盲盒体验。

业务中断类异常发现能力：自动扫描新增的微服务、web 接口、定时任务和依赖的中间件组件等，自动采集数据，并通过自动配置服务的断言信息发现业务异常

业务中断类异常响应能力：实时通报影响的用户旅程、影响用户的数量和影响的扩散速度，帮助用户快速判断当前问题的严重程度，选择合适的时间一键拉群、进行止血操作

V0.2

trace 智能对比模式：自动的对 Trace 数据、日志数据、指标数据、节点断言数据进行分析，让开发可以精准找到能够解决问题的人

V0.2.1

SRE 运营数据还原能力：

1. 还原核心用户旅程的不可用影响面及其变化趋势，体现质量治理的效果
2. 还原核心用户旅程的不可用事件的分布明细，辅助分析质量治理改进项

10 分钟玩转 V1.0 版本

使用场景

使用场景一：月度质量复盘大会前，通过质量运营数据，分析核心业务当月的不可用用户、不可用事件情况，与上月的对比；以及不可用事件的分布，与变更、发布、基础组件的关联，从而分析出下个月的稳定性治理改进项。

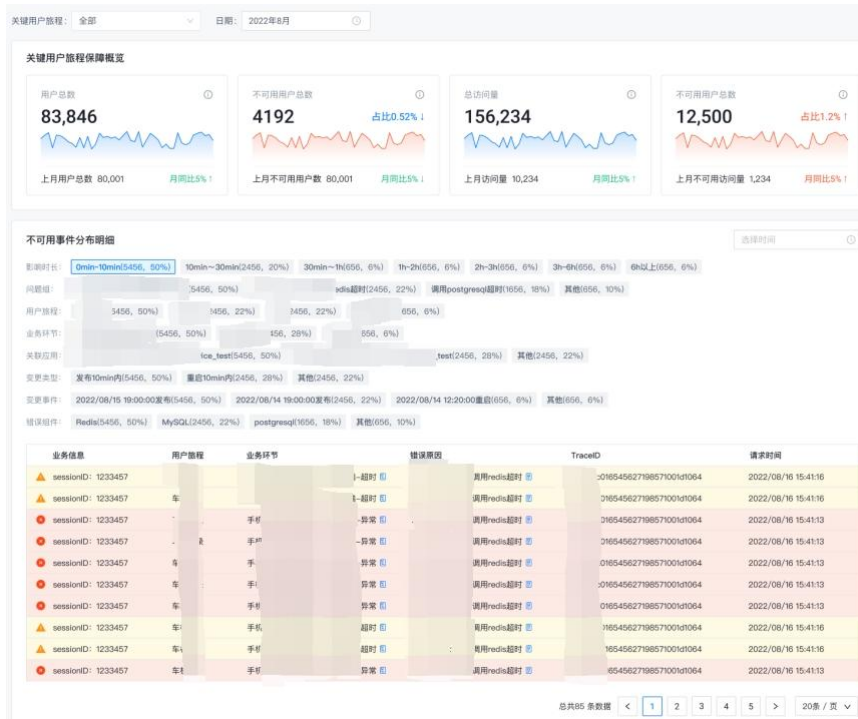
使用场景二：应用发布后，能够快速地发现服务异常、基础中间件异常类问题的失败情况，迅速处理

使用场景三：生产环境，SRE 在收到故障通知后，精准快速的定位服务异常、基础中间件异常类造成的问题

使用场景四：测试环境，开发连调时遇到失败/测试回归报错时，可以通过产品快速

定位到服务异常、基础中间件异常类造成失败的节点

稳定性质量运营



使用教程

1. 确定 CUJ 范围

在开始使用前，需要确认本次需要保障的关键用户旅程（简称：CUJ），以及该 CUJ 的各个业务环节的 API 入口。

示例：

3.新链路-生成二维码

4.新链路-手机端扫码

5.新链路-手机发起扫...

2. 安装 trace-agent

需要将步骤 1 中 CUJ 经过的应用全部接入 trace-agent，以支持探针采集 trace，上传到 Takin 平台进行 trace 日志数据分析。

◇ 请注意：

网关应用也需要接入探针

trace-agent 的接入方式详见：[AgentManagement 使用手册](#)

3. 业务还原

① 在对应的环境下，对 CUJ 的各个业务环节发起业务请求。

请求方式一：通过页面点击

请求方式二：通过脚本发起

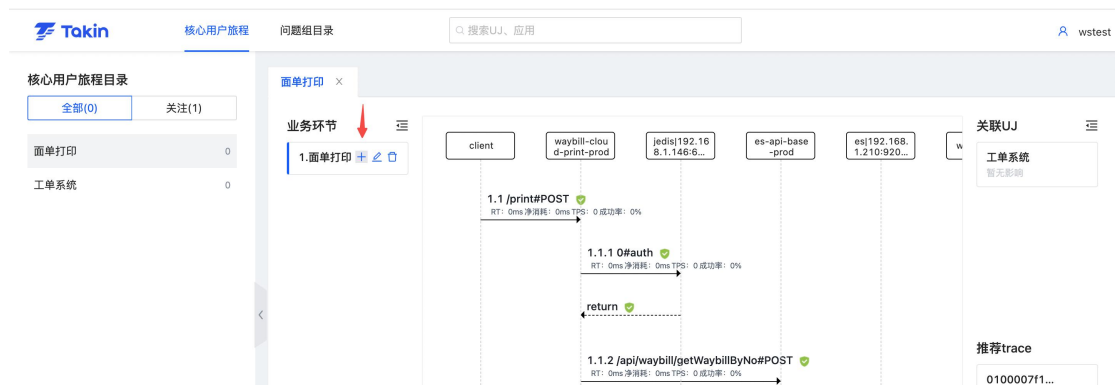
② 进入 Takin 平台：<http://sre.takin.shulie.io/home>



③CUJ 确认

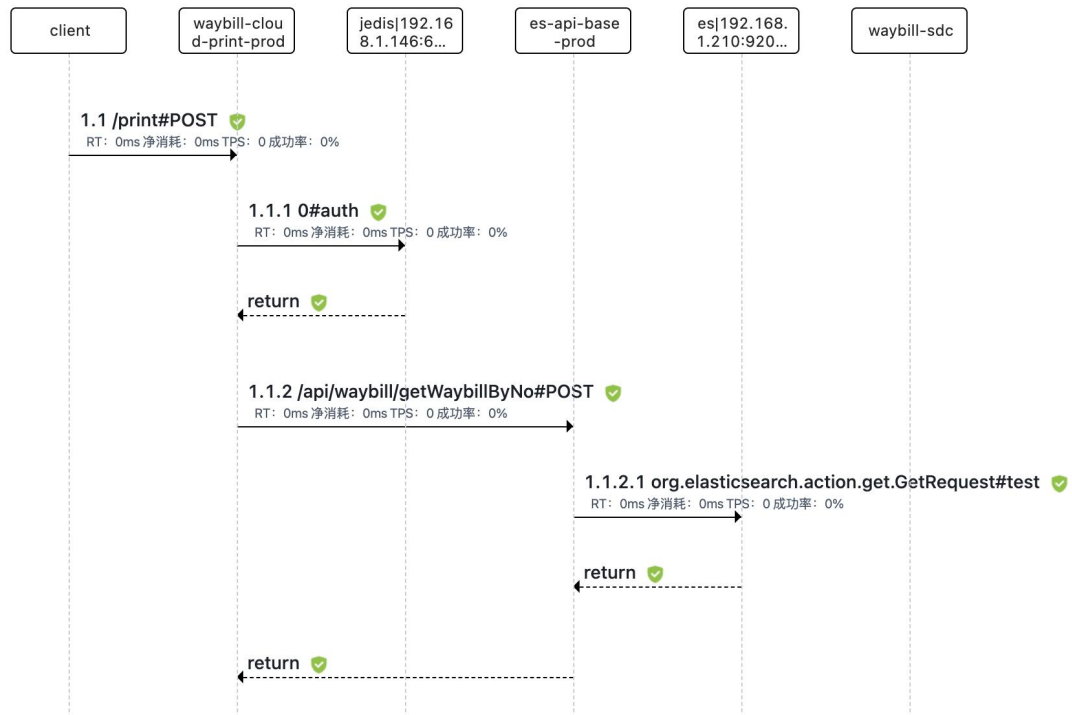
可以看到步骤①中 CUJ 各个业务环节已经被梳理出来，在左侧的 CUJ 列表。

这里可以先手工将几个业务环节添加到一个 CUJ 中。



④点击业务环节，可查看该业务环节的系统时序图，该时序图将该业务环节整条链路上的微服务应用、服务、中间件之间的关系按照调用的路径顺序还原了出来。

时序图(Sequence Diagram), 亦称为序列图、循序图或顺序图, 时序图描述了系统中类和类之间的交互, 将这些交互建模成消息交换, 时序图描述了类以及类之间的交互以完成的期望行为的消息。



业务健康实况还原

Takin 通过采集实时的请求 trace 数据, 加上 TestCase 的断言分析, 来实时监控业务的健康度。

业务时序图上的小盾牌, 代表了各个节点的 TestCase 的保障健康度。

TestCase 的断言, 目前支持系统异常和延时类。

基础系统异常包括:

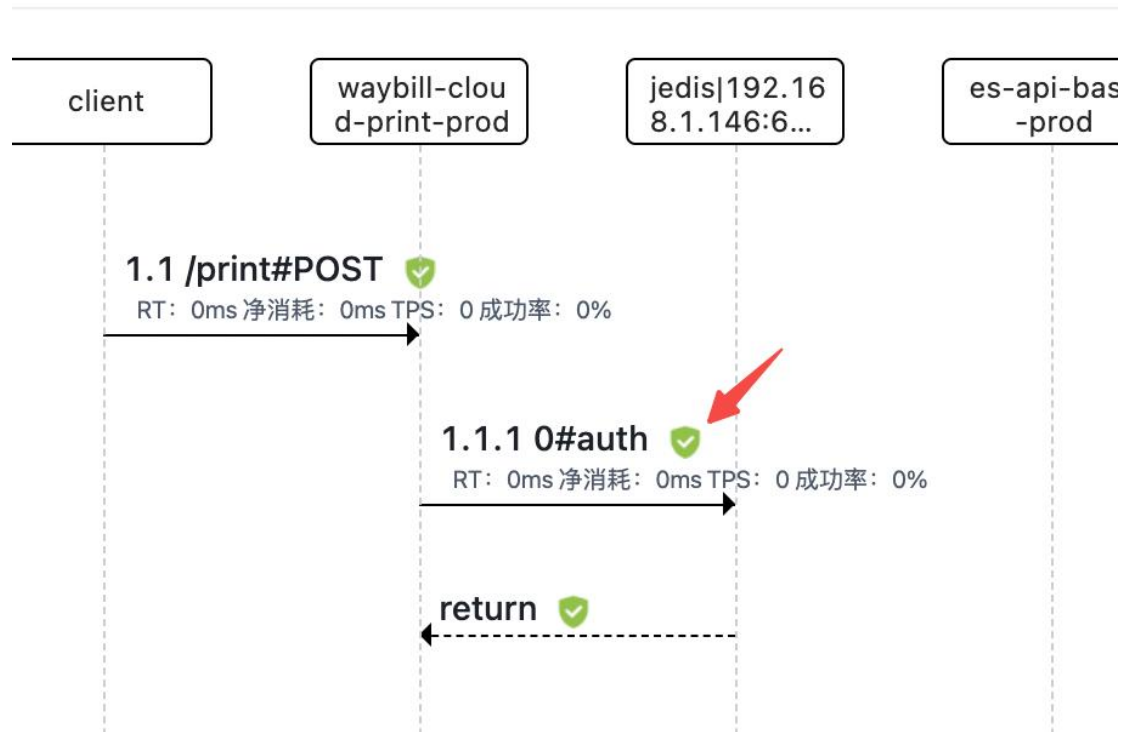
1. 入口 HTTP 服务的响应码异常 (≥ 400 的响应 code)
2. 微服务调用的异常
3. 各类中间件的请求异常

延时类异常包括:

1. 总耗时超过日基线
2. 服务自消耗超过日基线

正常状态下

TestCase 保障实况



trace 保障实况

业务信息	入口表现	错误原因	TraceID	时间
●	调用waybill-cloud-print-prod请求自耗时正常		0100002f16600443181302590d2a5f0001	2022-08-09 19:25:24
●	调用waybill-cloud-print-prod请求自耗时正常		0100002f16600443100502588d2a5f0001	2022-08-09 19:25:14
●	调用waybill-cloud-print-prod请求自耗时正常		0100002f16600443059932587d2a5f0001	2022-08-09 19:25:14
●	调用waybill-cloud-print-prod请求自耗时正常		0100002f16600442938722584d2a5f0001	2022-08-09 19:25:04
●	调用waybill-cloud-print-prod请求自耗时正常		0100002f16600442979102585d2a5f0001	2022-08-09 19:25:04

异常状态下

TestCase 运行实况

当有用户的请求出现异常/错误时，时序图上小盾牌会呈现红色

点击可查看 TestCase 的运行记录

绿色标识表示该次请求在本次服务节点的 TestCase 运行**成功**；

红色标识表示该次请求在本次服务节点的 TestCase 运行**错误**，且该次请求在请求入口处的 TestCase 也运行**错误**，表示**对用户造成了影响**；

黄色标识表示该次请求在本次服务节点的 TestCase 运行**错误**，但该次请求在请求

入口处的 TestCase 运行成功，表示对用户不一定造成了影响；

调用名: org.elasticsearch.action.get.GetRequest#test	TC 运行状况	TC 监控项	<< 切换到CUJ视角
调用waybill-cloud-print-prod请求自耗时正常	es-api-base-prod调用es-api-base-prod请求自耗时正常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时正常	es-api-base-prod调用es-api-base-prod请求总耗时正常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时正常	es-api-base-prod调用es-api-base-prod异常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时异常	waybill-cloud-print-prod调用es-api-base-prod请求自耗时异常	0100007f16600443140912589d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时正常	waybill-cloud-print-prod调用es-api-base-prod请求自耗时异常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时正常	waybill-cloud-print-prod调用es-api-base-prod请求总耗时异常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27
调用waybill-cloud-print-prod请求自耗时正常	waybill-cloud-print-prod调用es-api-base-prod正常	0100007f16600443181302590d2a5f0001	2022-08-09 19:25:27

trace 保障实况

保障实况	保障信息	时间选择: 最近30分钟 1小时 6小时 12小时 1天	2022-08-09 02:47 → 2022-08-10 00:47	实时更新: <input type="checkbox"/> 仅看失败
调用waybill-cloud-print-prod请求自耗时正常		0100007f16600426027632411d2a5f0001	2022-08-09 18:56:51	
调用waybill-cloud-print-prod请求自耗时正常		0100007f16600425987202410d2a5f0001	2022-08-09 18:56:51	
调用waybill-cloud-print-prod请求总耗时异常	es-api-base-prod调用es-api-base-prod异常	0100007f1660038308352405d2a5f0001	2022-08-09 17:45:21	
调用waybill-cloud-print-prod请求总耗时异常	调用waybill-cloud-print-prod请求总耗时异常	0100007f16600383123952406d2a5f0001	2022-08-09 17:45:21	
调用waybill-cloud-print-prod请求自耗时异常	waybill-cloud-print-prod调用waybill-cloud-print-prod异常	0100007f16600383002652403d2a5f0001	2022-08-09 17:45:11	

4. 问题定位

观测到有异常的 trace 后，可通过 trace 的智能对比，排查问题点。

点击 traceID，进入 trace 智能对比页面。

系统会默认选取一条完全正常的 trace 作为基准，将当前的 trace 与基准 trace 进行智能对比。

观测差异点

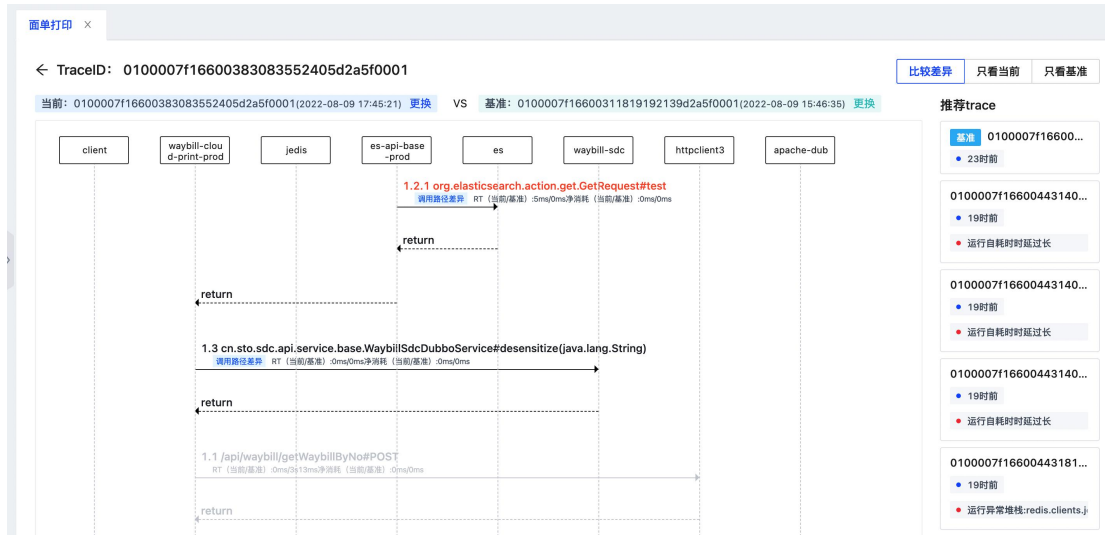
智能对比包括三方面：

1. trace 调用路径对比

- 调用路径差异：当前 trace 调用了该服务，基准 trace 未调用；
- 灰色部分：当前 trace 未调用，基准 trace 调用了该服务；

2. 服务响应 code 对比

3. TestCase 结果对比



对比差异点详情

观测到差异点后，可点击差异点查看请求节点的请求地址、请求 header、请求参数、响应结果、请求客户端、服务端应用的业务日志，通过各类数据的对比、分析找到出问题的点和能解决问题的责任人。

响应对比

服务: org.elasticsearch.action.get.GetRequest#test | 请求响应 | 业务日志

响应 | 请求头 | 请求体

当前Trace (httpCode: 01,请求方法: test,请求地址: 192.168.1.100:9200:0org.elasticsearch.action.get.GetRequest)

```

1- java.net.ConnectException: 拒绝连接
2- at org.elasticsearch.client.RestClient.extractAndWrapCause(RestClient.java:804)
3- at org.elasticsearch.client.RestClient.performRequest(RestClient.java:225)
4- at org.elasticsearch.client.RestClient.performRequest(RestClient.java:212)
5- at org.elasticsearch.client.RestHighLevelClient.internalPerformRequest(RestHighLevelClient.java:699)
6- at org.elasticsearch.client.RestHighLevelClient.performRequest(RestHighLevelClient.java:699)
7- at org.elasticsearch.client.RestHighLevelClient.performRequestAndParseEntity(RestHighLevelClient.java:699)
8- at org.elasticsearch.client.RestHighLevelClient.get(RestHighLevelClient.java:699)
9- at io.shulie.demo.es.controller.IndexController.queryByNo(IndexController.java:133)

```

基准Trace (httpCode: ,请求方法: ,请求地址:)

header 头对比

服务: /print#POST | 请求响应 | 业务日志

响应 | 请求头 | 请求体

当前Trace (httpCode: 200,请求方法: POST,请求地址: 127.0.0.1:32846/print)

```

1 {
2   "referer": "http://192.168.1.28/tro/",
3   "host": "localhost:9008",
4   "user-agent": "curl/7.29.0",
5   "accept": "*/*"
6 }

```

基准Trace (httpCode: 200,请求方法: POST,请求地址: 127.0.0.1:64593/print)

```

1 {
2   "referer": "http://192.168.1.28/tro/",
3   "host": "localhost:9008",
4   "user-agent": "curl/7.29.0",
5   "accept": "*/*"
6 }

```

请求体对比

服务: /print#POST | 请求响应 | 业务日志

响应 | 请求头 | 请求体

当前Trace (httpCode: 200,请求方法: POST,请求地址: 127.0.0.1:32846/print)

```

1 {
2   "httpPrms": {
3     "mailNo": [
4       "OnFyPY18db7iS1wt0j_"
5     ],
6     "index": [
7       "test"
8     ]
9   }
10 }

```

基准Trace (httpCode: 200,请求方法: POST,请求地址: 127.0.0.1:64593/print)

```

1 {
2   "httpPrms": {
3     "mailNo": [
4       "OnFyPY18db7iS1wt0j_"
5     ],
6     "index": [
7       "test"
8     ]
9   }
10 }

```

业务日志分析



5. 风险运营

进入「[风险运营](#)」页面，可以看到各个关键用户旅程内存在的风险。



新增风险

点击新增风险，可以添加找到的问题



编辑风险

选择需要确认/处理的风险，点击编辑，可以查看风险详细信息；

风险确认后，可更新为「待处理」状态，表示风险已采纳，需要排期处理解决

风险有新的处理动态，可对风险详情进行编辑

风险解决后，可更新为「待验证」状态，表示风险已解决，需要进行验证

风险验证通过后，可更新为「已关闭」状态，表示风险已关闭

6. 事件分析

进入「事件查询」页，可以以多维度搜索用户事件

其中，关键字搜索支持 header 头、请求体、响应体里的 value 值匹配

业务信息	关键用户旅程	业务环节	入口	入口表现	请求时间	操作
影响用户:[gder92200001113]、请求:010a1eac16643659750941379d25991	工单系统11	-	/query/workorderinfo#POST	调用st-woc_test失败	2022-09-28 19:52:55	查看trace
影响用户:[gder92200001113]、请求:010a1eac16643572824489954d25991	工单系统11	-	/query/workorderinfo#POST	调用st-woc_test失败	2022-09-28 17:28:02	查看trace
影响用户:[gder92200001113]、请求:010a1eac16643458631058037d25991	工单系统11	-	/query/workorderinfo#POST	调用st-woc_test失败	2022-09-28 14:17:43	查看trace

FAQ

Q：数据保存多久

数据保留策略如下：

正常事件用于指标计算后，选择性留存几条正常数据，其他的正常数据自动清理，仅保留 1 天的时间。

错误事件的事件日志会继续留存，以供后续的问题排查分析。