SRE-Copilot：大语言模型与aiops结合的探索

张翔　字节跳动

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

上海站
K+全球软件研发行业创新峰会
时间：2024.06.21-22

敦煌站
K+思考周®研习社
时间：2024.10.17-19

香港站
K+思考周®研习社
时间：2024.11.10-12

K+峰会详情

AiDD峰会

上海站
AI+研发数字峰会
时间：2024.05.17-18

北京站
AI+研发数字峰会
时间：2024.08.16-17

深圳站
AI+研发数字峰会
时间：2024.11.08-09

AiDD峰会详情

演讲嘉宾

张翔

字节跳动SRE-Copilot负责人

中科院计算所博士毕业，字节跳动基础架构SRE数据化方向负责人，聚焦成本、稳定性、效率、服务四条主线，为SRE提供数据化与智能化支持。加入字节后，主导了异常检测、智能变更、故障诊断、智能限流、运筹优化、大语言模型应用、资源交付数据化运营、运维数仓等多个数智化运维项目的上线与推广。

目录
CONTENTS

PART 01
SRE-Copilot整体架构

▶AI-Agent相关概念：Tool calling

Function calling是可靠地将LLMs连接到外部工具以实现有效的工具使用和与外部API的交互的能力。

“今天天气怎么样？”
再智能的大模型对这个问题也束手无策

```
messages = [
    {
        "role": "user",
        "content": "What is the weather like in London?"
    }
]
```

```
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_current_weather",
            "description": "Get the current weather in a given location",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "The city and state, e.g. San Francisco, CA",
                    },
                    "unit": {
                        "type": "string",
                        "enum": ["celsius", "fahrenheit"]},
                },
                "required": ["location"],
            },
        },
    }
]
```

```
ChatCompletionMessage(content=None, role='assistant', function_call=None, tool_calls=
[ChatCompletionMessageToolCall(id='...',
function=Function(arguments='{"location":"London","unit":"celsius"}', name='get_current_weather'),
type='function')])
```

AI-Agent相关概念： RAG

检索增强生成（Retrieval-Augmented Generation，又称RAG）通过检索LLMs之外的数据源来支持其生成答案。RAG=搜索+LLM提示，根据用户的查询要求，LLMs会使用搜索算法从外部数据源获取上下文信息，最后，查询和检索到的上下文合成后送入到LLM的提示中。

私域知识

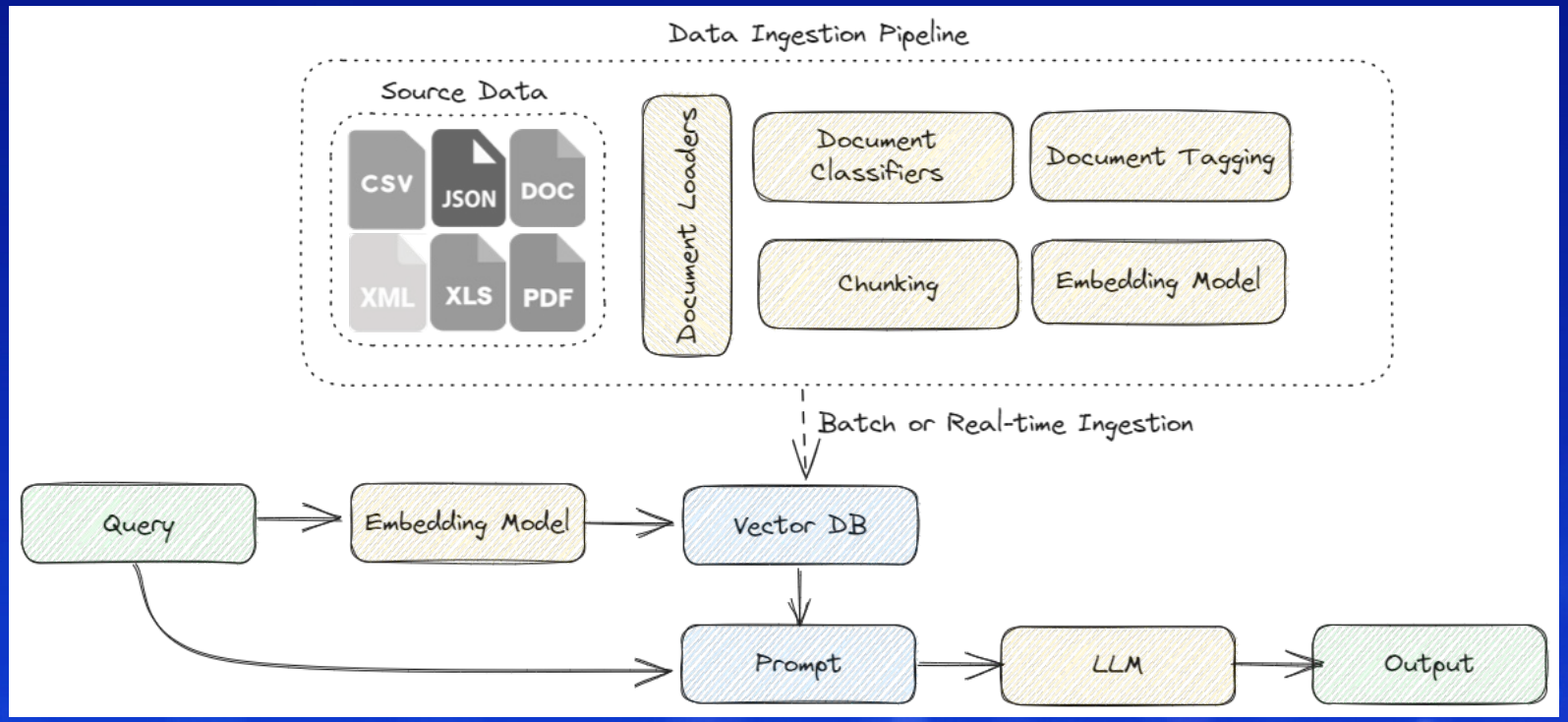
“公司服务器的数量，线上的利用率是多少”
“当前工单状态是什么”

新知识问题

“今天有哪些故障？”
“最新电影的主演是谁”

长尾问题

“使用XX语言在某环境如何连接XX数据库”



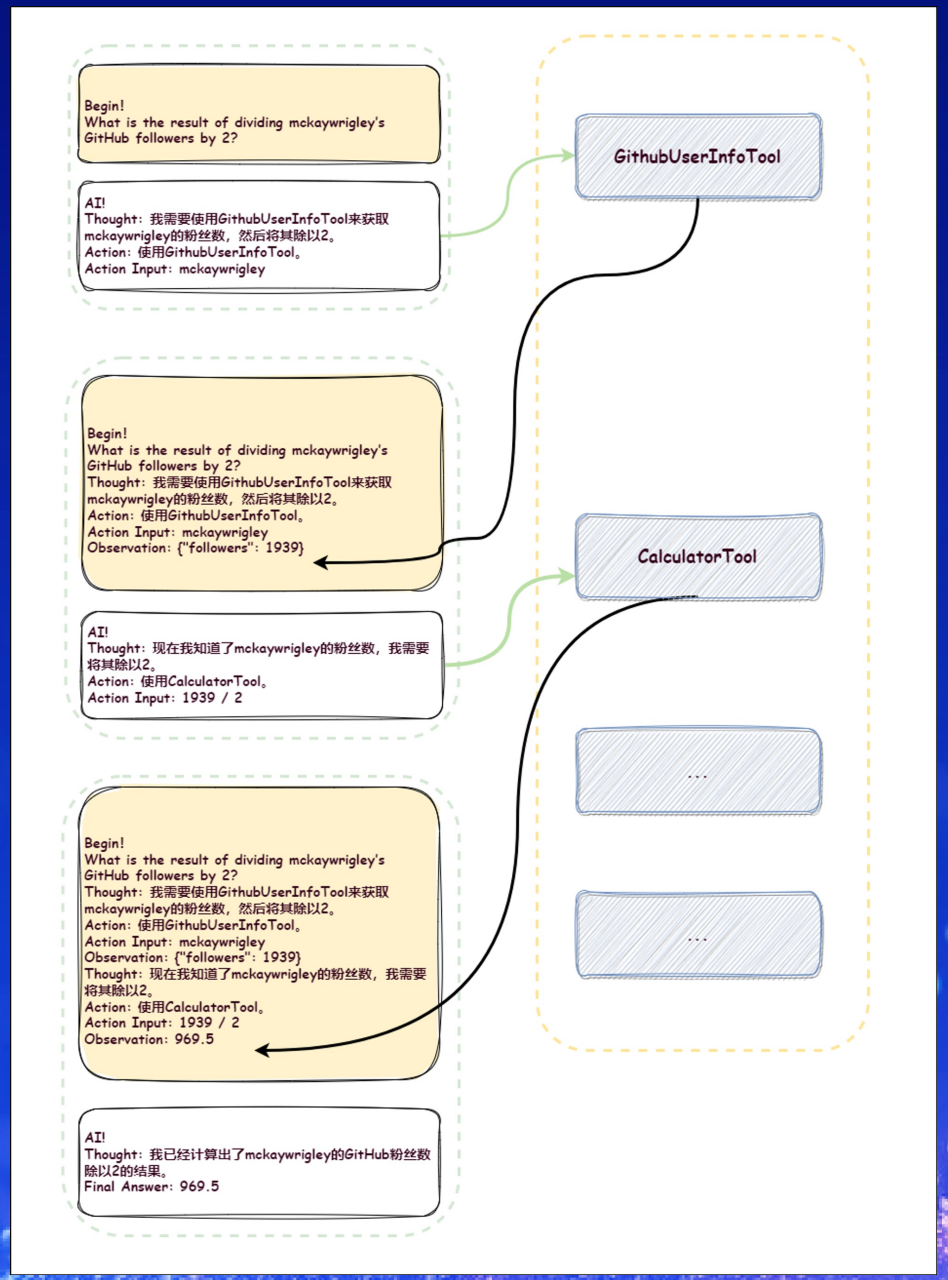
AI-Agent相关概念：Reason + Act

ReAct是一种LLM提示和结果处理方法，它结合了推理、行动规划和知识来源的整合，使LLM超越其语言模型，在其预测中使用来自现实世界的信息。ReAct是推理和行动的结合。

(2b) ReAct (Reason + Act)

```
Act 1: Think[First I need to find a pepper shaker...more
likely to appear in cabinets (1-6), countertops (1-3), ...   ]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1,
...... a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to
put it in/on drawer 1. ]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 …

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on drawer 1
```

步，或者任务复杂结果充满不确定时，rea

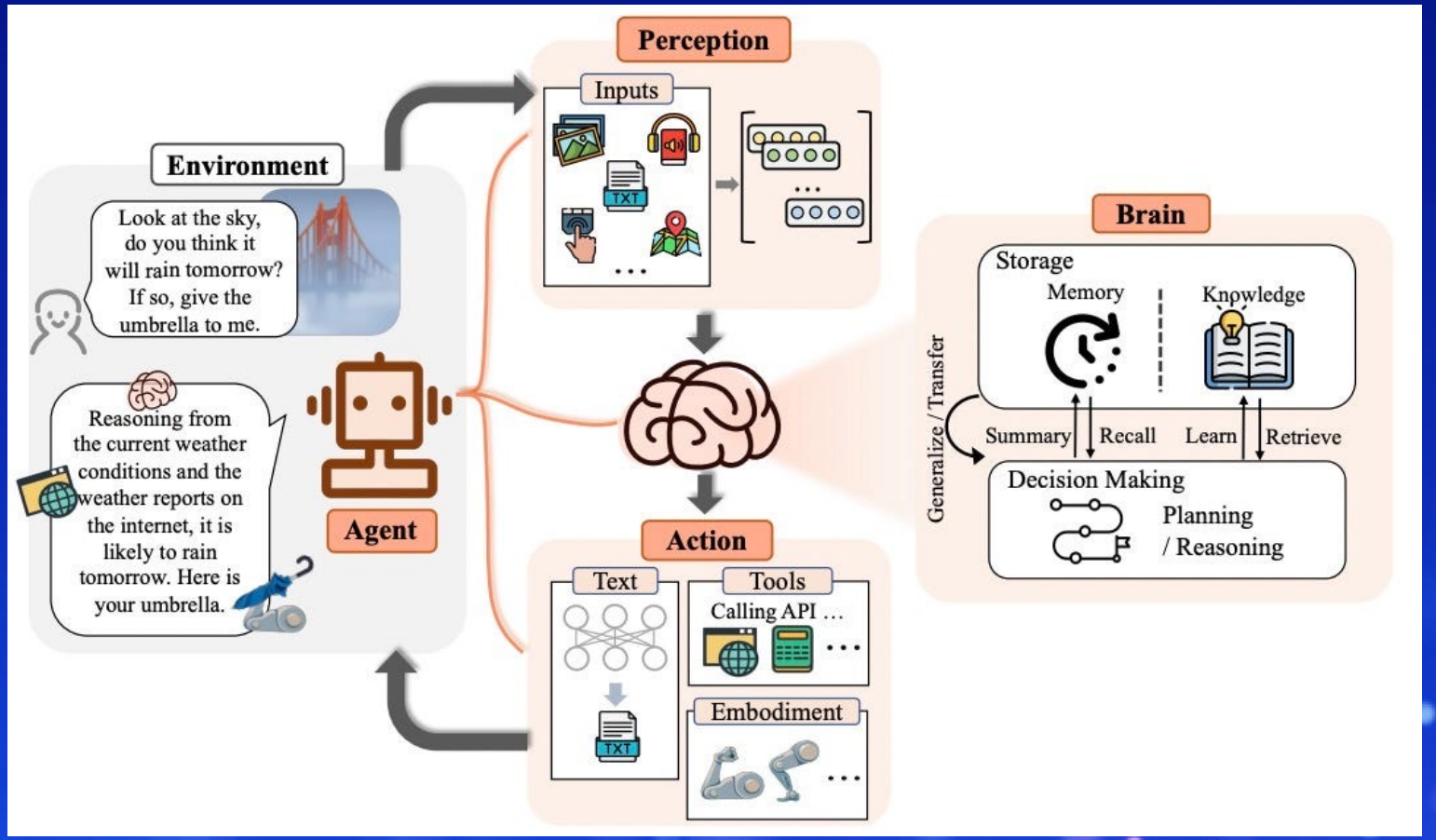


AI-Agent相关概念—Agent智能体

代理（Agent）指能自主感知环境并采取行动实现目标的智能体。

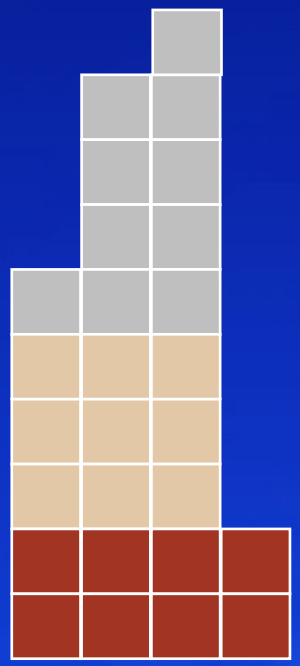
AI代理的整体框架由三个关键部分组成：大脑、感知和行动。

- 大脑：大脑主要由一个大型语言模型组成，不仅存储知识和记忆，还承担信息处理和决策功能，并能呈现推理和规划过程以处理未知任务。
- 感知：感知模块的核心目的是将代理的感知空间从纯文本领域扩展到包括文本、听觉和视觉模态。
- 行动：在代理的构建中，行动模块接收大脑模块发送的行动序列，并执行与环境交互的行动。



SRE-Copilot整体架构

SRE-Copilot是基于LLM的多场景智能运维框架，支持Multi-Agent协作与动态编排，具备计划、记忆、反思、推理与ReAct等能力，为SRE提供智能化服务。



Copilot

意图识别，参数提取，将任务调度分配给合适的子Agent

- LogAgent
- TraceAgent
- RCAAgent
- TradeAgent
- MonitorAgent
- CMDBAgent
- ...

多源数据Agent：对各种数据源进行异常检测，返回故障描述

- QAAgent
- WorkflowAgent
- ReportAgent
- CodeAgent

功能型Agent：知识库问答，工作流规划，写报告，写代码等功能

参考GPT的思想，通过集成学习多个专业的LLM的agent组成强大的混合专家*(MoE，Mixture of Experts)*系统。

效果展示


Oncall Agents
MonitorAgent
CMDBAgent
TraceAgent
LogAgent
TradeAgent
Copilot
RCAAgent
SRE-COPILOT
SRE-Copilot是基于LLM的多场景智能运维框架，支持Mulit-Agent协作与动态编排，具备计划、记忆、反思、推理与ReAct等能力，为SRE提供智能化服务。
有什么问题吗?

PART 02
点期望解决的运维痛

SRE-Copilot 优势

痛点	传统AIOPS	SRE-Copilot
系统复杂，依赖繁多，海量数据	单个运维专家，甚至单个团队难以掌握上下游全部知识和技术细节，也难以处理全部告警/异常	大模型几乎能学习人类全部知识，通过Multi-Agent以及知识库可以无限扩展
数据无标注 训练成本高	大部分异常检测算法依赖标注，无监督算法能力一般， 根因诊断算法更加依赖标注。专家经验很难编码成算法模型	把专家经验经验转化为故障表现，让模型推断，无需训练
接入/维护成本	要完全理解每一个复杂模型，维护门槛高；客户的数据和系统都是私域的，需为客户现场定制与优化，增加了接入成本。调整或接入新数据要重新训练	通过“混合专家模型”的集成学习概念，只需关注组件与模型，客户自己的模型/逻辑也可以像乐高积木一样轻松接入，灵活调整，甚至框架自己可以动态编排
新的故障推理	无法推理未知故障	LLM已经出现了涌现和推理能力，基于自己的通用知识，并且可以不断学习领域知识进行推理，似乎是解决新故障根因定位的最佳选择
交互使用成本	交互复杂，需要严格传递参数等	自然语言交互，更加智能，可以开放给更多用户

PART 03
框架实现的技术细节

异常检测——基于ReAct框架与思维链CoT的Multi-Agent编排

角色定义

Copilot主持人：解析用户需求，制定运维Plan，安排不同Agent工作（如根因定位交给RCA）

多数据源Agent：分别负责不同模态的数据，选择合适的算法进行异常检测与检索

RCAAgent：收集其他Agent检测到的异常信息与链路、配置信息，进行根因定位

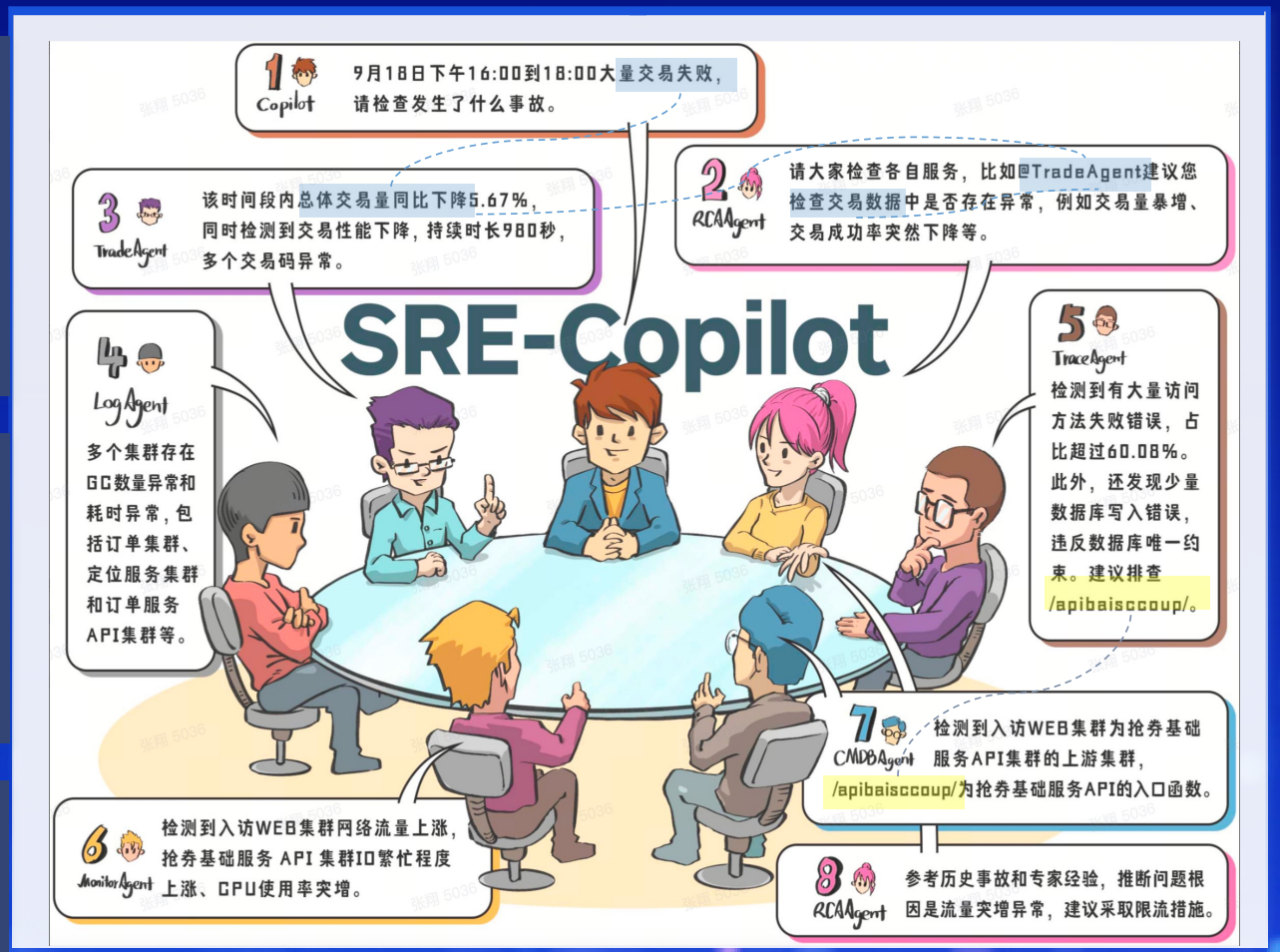
Multi-Agent编排

ReAct包括推理Reasoning和行动Action，推理帮助模型生成、追踪和更新计划并处理异常，行动允许模型与外部环境交互以获取更多信息Observation，提升准确率与适应性

每个Agent均根据检测到的异常动态编排，RCAAgent负责收敛协作轮次，并根据其他Agent反馈决定下一步分析与下钻的方向

优势

拟解决真实云平台跨组件协同定位问题，可以采用多个agent模拟多个组件运维团队



根因定位 ——基于RAG增强的推理与反思

本次故障持续10分钟，CPU飙升，内存打满，接口出现大量失败.....

向量数据库

专家经验：内存打满后服务一般不可用，此时应重启.....

历史故障：2023年11月5日，xx接口不可用，持续半小时， CPU飙升.....

专家经验：.....

历史故障：.....

Knowledge 本次故障.....

专家经验：.....

专家经验：.....

历史故障：.....

Question

大语言模型

根因：内存打满

建议：重启

应用专家经验

- RAG检索增强：检索相关/支撑文档，作为上下文和原始提示词组合，再提交给LLM使用。LLM的知识是静态的，且会有“幻觉”现象，RAG让LLM不用重新训练就能获取最新相关信息或者私域知识
- 我们用RAG检索最符合本次故障的Top N专家经验，交由LLM进行根因定位

学习新的故障

- 每次诊断结果会加入模型记忆，再次诊断时对最相近专家经验与诊断结果进行推理，让模型获得持续学习的能力
- 反思 Reflexion：通过让模型进行自我评估和自我反省，能够进一步提升模型推理根因的准确率30%以上

推理未知故障

- 对于未发生过的，也没有专家经验的新故障，SRE-Copilot也会根据自己的知识，尝试推理故障根因
- 我们在一些新故障上发现模型的推理能力，比如在没有经验和历史的情况下，正确诊断出磁盘写满/GC等故障

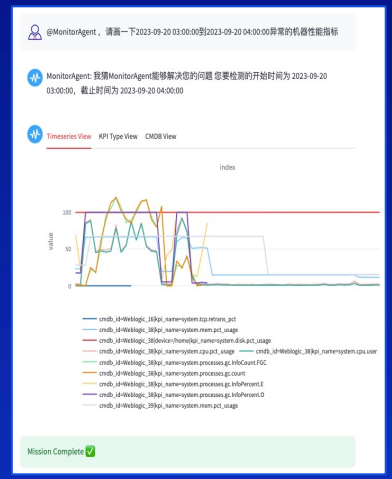
基于function-call的多功能实现

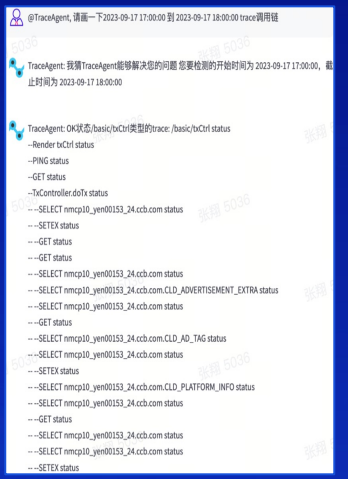
LLM优势：收敛模型统一入口，理解、拆分用户意图，编排调用不同工具，实现多场景复杂智能运维，并持续学习迭代

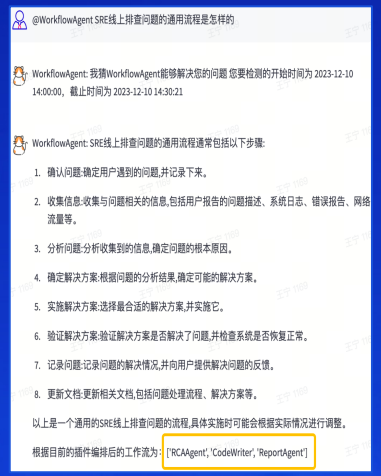
故障分类

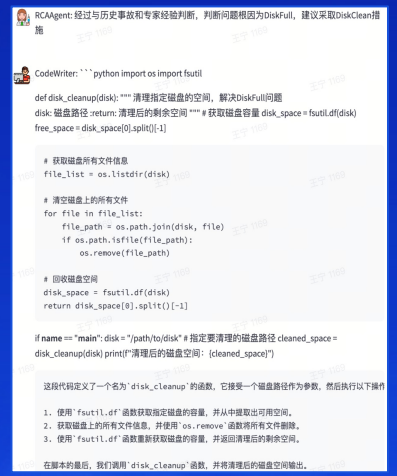
故障类别	解释说明
流量异常	用户入流流量突增异常，一般建议进行限流
磁盘空间不足	磁盘被写满，一般建议清理冗余文件
CPU利用率过高	无状态服务的CPU利用率突增，一般建议重启处理
JAVA进程GC异常	GC次数过多，导致程序性能下降或崩溃，一般建议调整GC参数

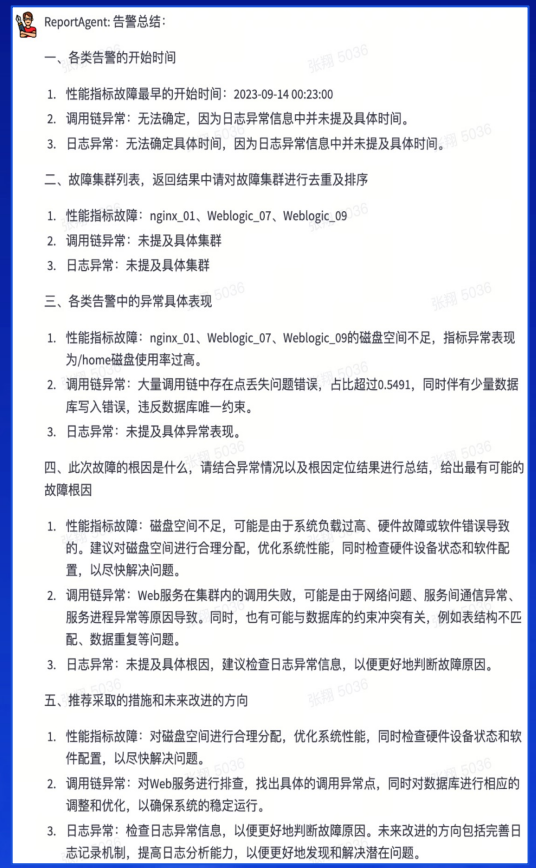
运维可视化

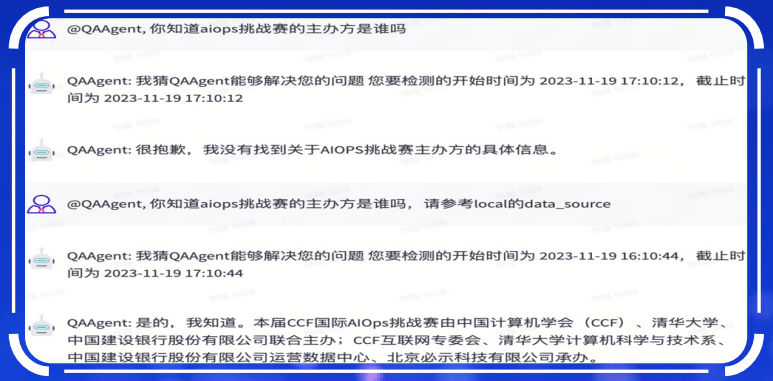












工作流生成（运维计划）　故障自愈+代码生成　故障报告　知识库问答

PART 04
在字节跳动的应用场景

基于LLM的RCA-Agent构建

我们优先在根因诊断的场景探索Agent的模式

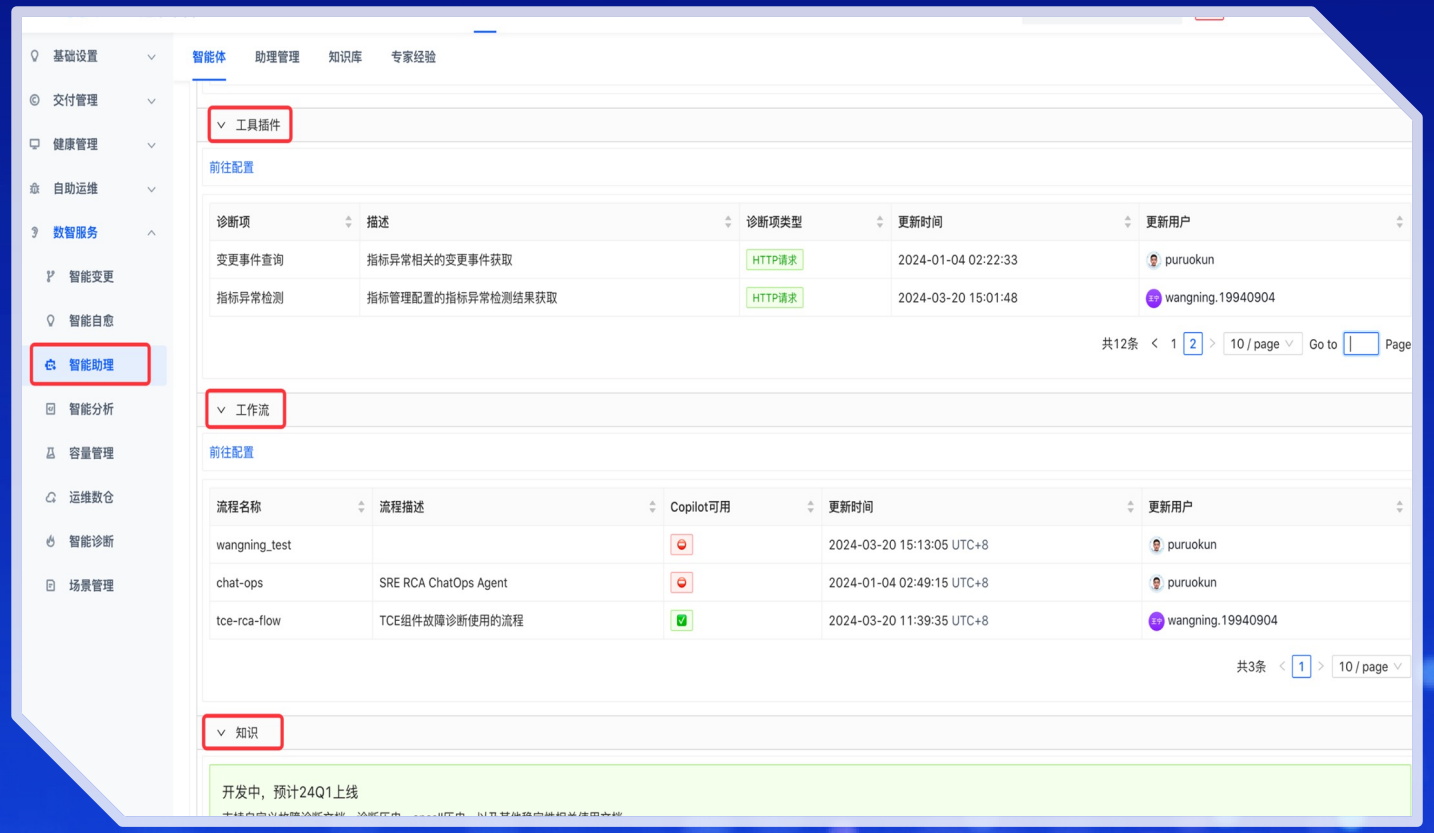
01 知识库的构建

02 基础工具的构建

03 核心工具：LLM根因推理

04 工作流的构建

05 Agent的使用场景

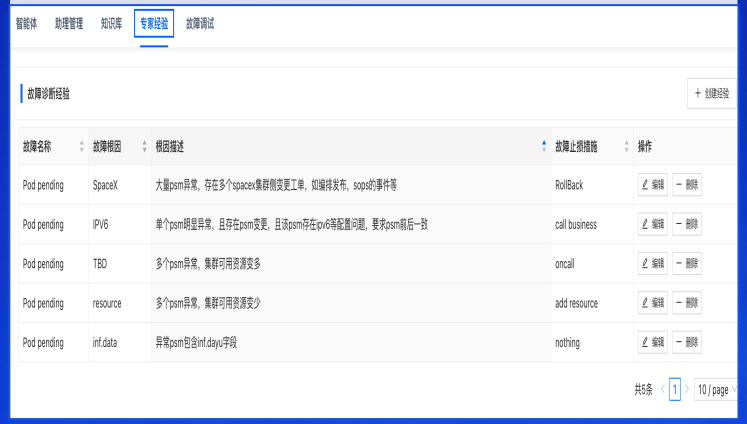


知识库的构建

知识库当前包含以下三个部分，后续我们还会持续引入用户文档，历史oncall等数据分块的数据

1.排障的专家经验

- 这里是只针对根因定位场景，业务同学可以将他们的经验积累&管理起来
- 我们定义专家经验是一组故障根因+故障表现+故障止损措施的组合，以便大模型去推理故障



2.故障场景的SOP文档

- 通过sop文档的形式，希望能提供给组件同学更加灵活的知识管理方式。当前由于大模型的能力局限，我们通过这种半规范的文档，将指标/诊断项、诊断流程等内容管理维护起来

3.历史的故障信息

- 每一次的历史故障，会被记录下来，用来给组件同学“训练”/打标模型

基础工具的构建

参考openai的tools/gpts的接入方式，我们将运维场景的指标和其他基础工具管理起来。
基础工具包含几类，例如指标通用的异常检测、变更事件查询、组件自定义的检测项等。

1.一些集群诊断场景的指标 2.自定义的检测项

Metrics指标

metrics指标名	监控部署区域	指标描述	核心tags	异常时下钻	结果描述	指标链接
add_on_controller.prod.cluster_metrics.pod_count	China-North	集群中podpending的数量	• reason=ResourcePending • namespace=default • status=Pending • cluster=${cluster} • node_level=${nodeLevel} • psm=*	psm	反映集群当前的Pending Pod数量，常态下应 < 10 指标上升：当前集群的Pending Pod增加，需要尽快排查和干预 指标下降：当前集群的Pending Pod降低，需要持续关注，直至降低为0 智能诊断需求：需要能够分析造成指标上涨的具体PSM，基于分析出的PSM进一步分析该PSM Pending的原因（通过抽样部分Pod，分析其Pod Event）	https://metrics-max:add_on_co=literal_or(*),pol_or(ResourceP

Influxdb指标

influxdb指标	指标描述	核心tags	异常时下钻	结果描述	dbname	InfluxQL
escloud_es_pod_cpu_metric	CPU核数使用率	node	node	• 指标上连续5分钟超过85%：CPU使用率过高 • 智能诊断需求：返回使用率过高的node列表，进一步排查是机器问题还是业务问题	xxx	SELECT max("value") FROM "escloud_es_pod_/^$escluster/ AND $timeFilter GROUP BY time(

诊断项名称	接口url	接口描述	参数	返回结果	调用示例	数据处理方法

集群状态查询	https://xxx.byted.org/api/v1/cluster/	查询集群状态	请求方式：get
			body：无
			鉴权：无
			params：{"cluster": "$cluster_name", "time_range": "2024-01-09%2017:20:00,2024-01-09%2017:40:00"}
				{
				"success": true,
				"data": {
				"cluster": "xxx",
				"region": "cn-beijing",
				"client_status": [
				{
				"status": "NO",
				"name": "延迟过大",
				"result": "延迟过大，延迟为2.8s",
				"reason": ""
				},
					curl --location 'https://xxx.byted.org/api/v1/cluster/?cluster=$cluster_name&time_range=2024-01-09%2017:20:00,2024-01-09%2017:40:00'	需要返回“client_status”列表中，所有status为“NO”的条目的“result”字段作为当前集群的异常表现
						let raw_res = Results.JSONBody();
						let res = filter(raw_res, .status == "NO");
						map(res, {.result})

用户实际部署的工具包含通用指标的异常检测、变更事件查询、自然语言的意图理解、大语言模型的根因推理

诊断项 | 诊断流程 | 通用诊断项 | 事件路由

新建诊断项

ID	诊断项名称	描述	诊断项类型	创建时间	更新时间	更新用户	操作
79	llm_infer	大语言模型推理根因	HTTP请求	2024-03-20 14:42:32	2024-03-20 16:45:33		编辑 删除
78	tce_platform_query	tce平台变更查询	HTTP请求	2024-03-20 11:59:30	2024-03-20 15:51:02		编辑 删除
77	spacex_change_query	查询spacex平台的变更	HTTP请求	2024-03-20 10:39:32	2024-03-20 16:29:06		编辑 删除
46	bke_prober	写入过程状态	规则组	2024-01-31 22:16:52	2024-01-31 22:16:52		编辑 删除
16	chat_ops	自然语言诊断任务指令机器人	HTTP请求	2024-01-04 02:47:58	2024-01-17 14:35:57		编辑 删除
15	tce_dignosis_rule_set	TCE常用诊断规则	规则组	2024-01-04 02:29:03	2024-01-04 02:29:03		编辑 删除
14	event_fetch	指标异常相关的变更事件获取	HTTP请求	2024-01-04 02:22:33	2024-01-04 02:22:33		编辑 删除
13	anomaly_detection	指标管理配置的指标异常检测结果获取	HTTP请求	2024-01-04 02:16:30	2024-03-20 15:01:48		编辑 删除

核心工具：LLM根因推理

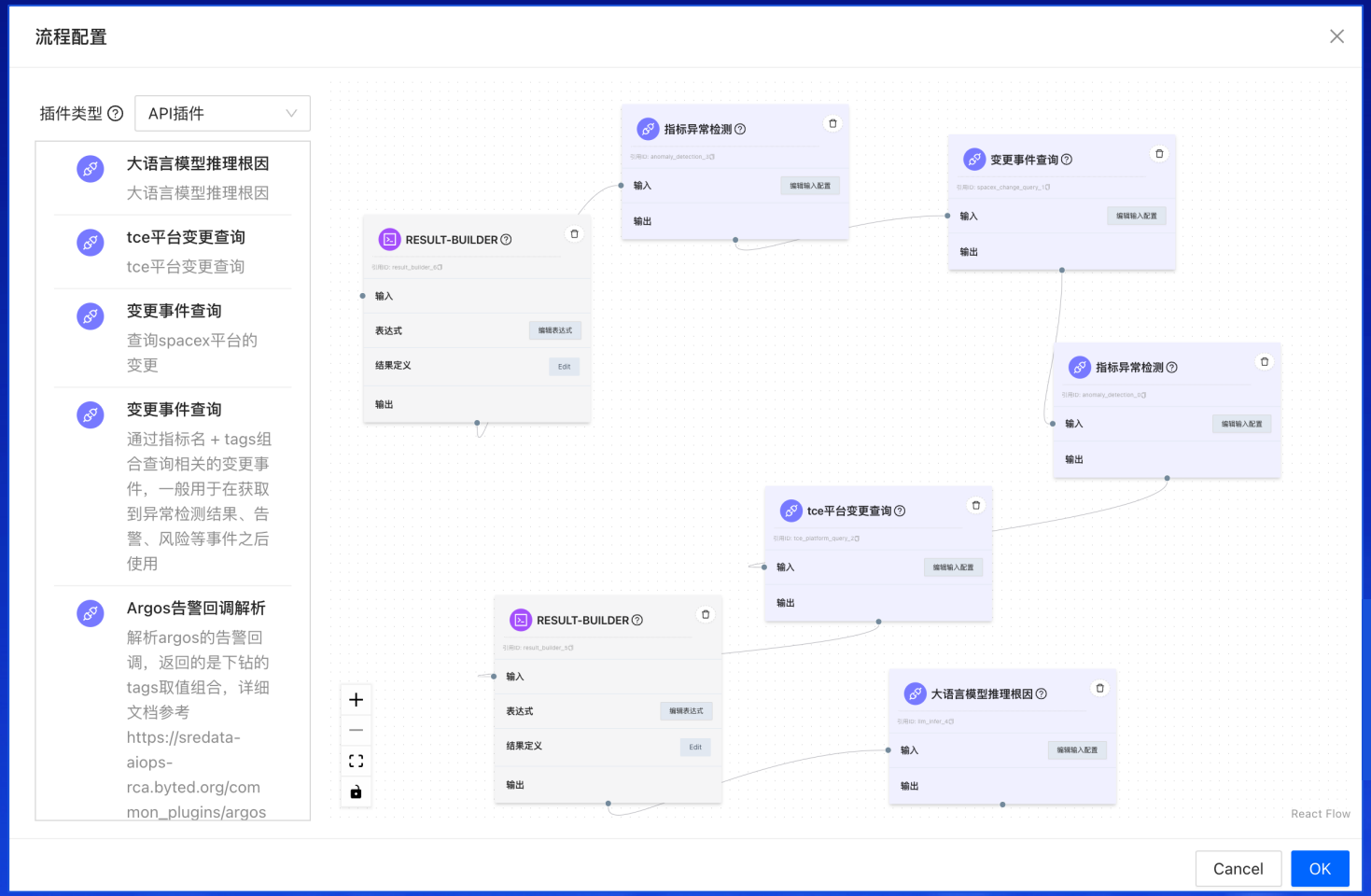
- 相较于传统的根因定位/故障分类
  ----将异常时刻的时序信息等进行编码聚类，在向量空间里面求距离和相似度进行分类
- 我们尝试对异常时刻的信息映射到自然语言描述，利用大语言模型的能力进行分类

展示了一个将诊断记录和专家经验整合的例子

用户可以调整专家经验、结果、prompt模板等，对模型进行测试

工作流的构建

根据上述提到的基础工具，用户可以自由组合这个工作流



- 目前仍然依赖用户自己配置
- 同时在探索让模型自己从用户的SOP文档中生成工作流，并利用ReAct的思路动态编排这些诊断项

Agent的应用与调试

引入LLM，改变了传统的交互方式，可以通过模糊地提问，识别意图和参数，并将结果优化展示后返回

当前的一些使用场景1

在群里/私聊机器人，用自然语言提问，会匹配你想要的工作流

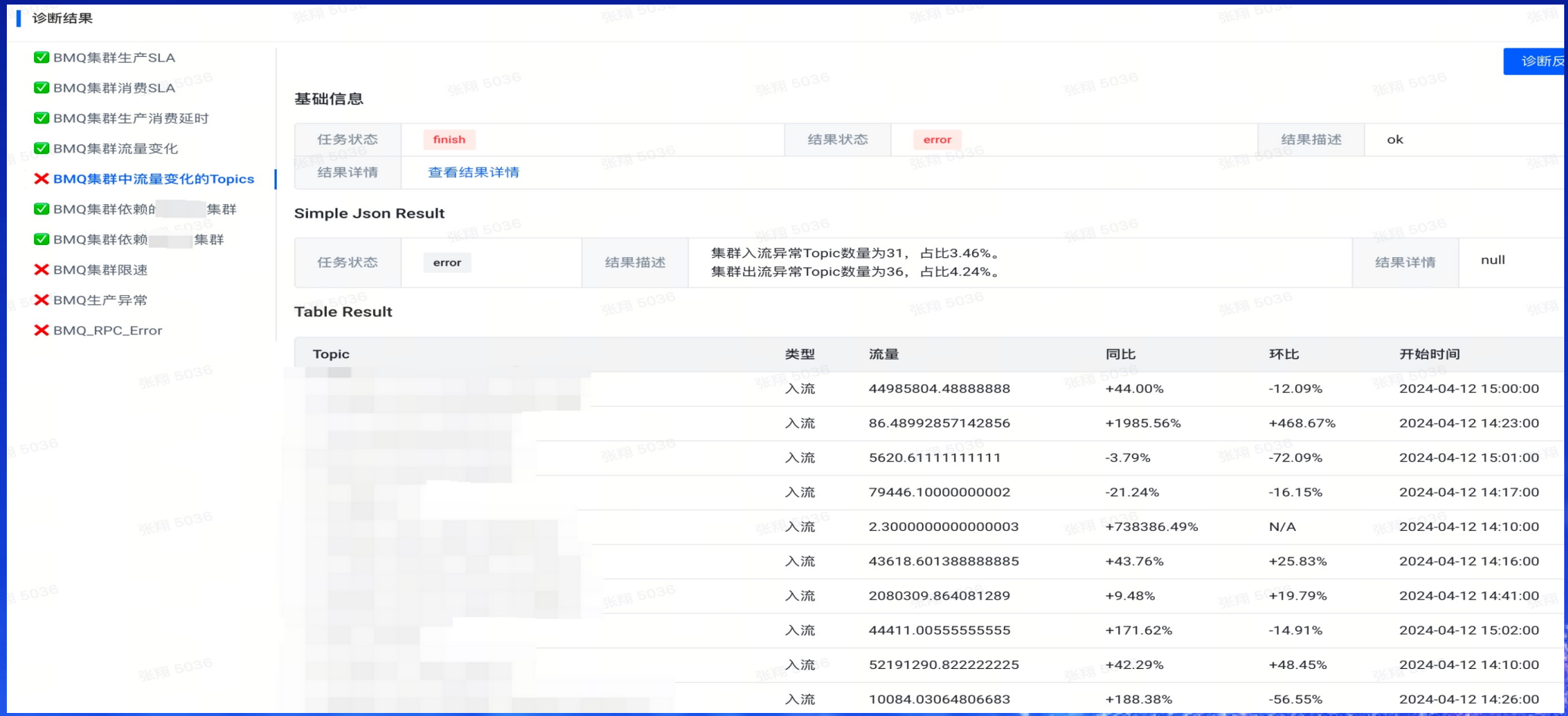
当前的一些使用场景2

事故拉群等场景自动匹配工作流，
对聊天记录里的关键信息提取，进行诊断

Agent的应用与调试

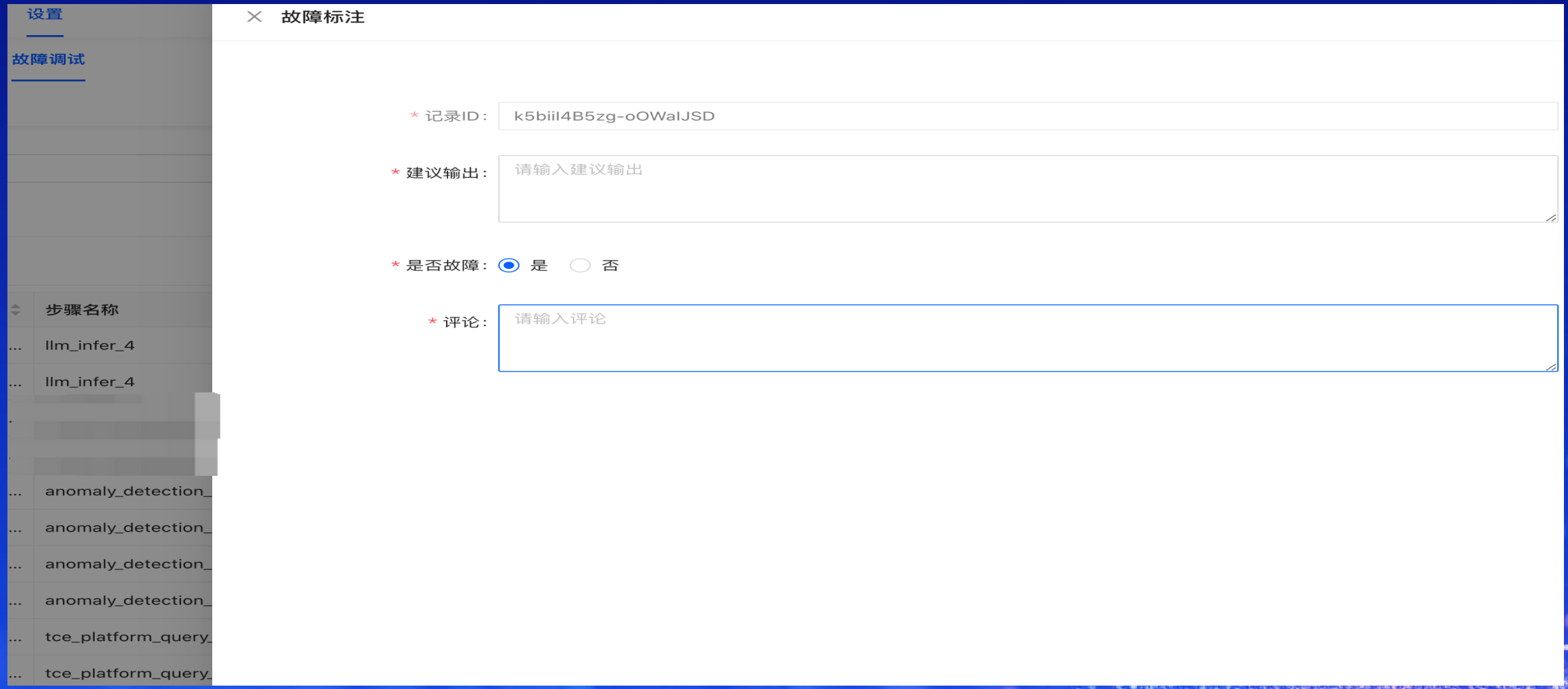
统一运维平台的各运维分站的诊断服务，后端也是SRE-Copilot实现的

- 告警绑定诊断 & 页面输入诊断
- 可设置自愈措施



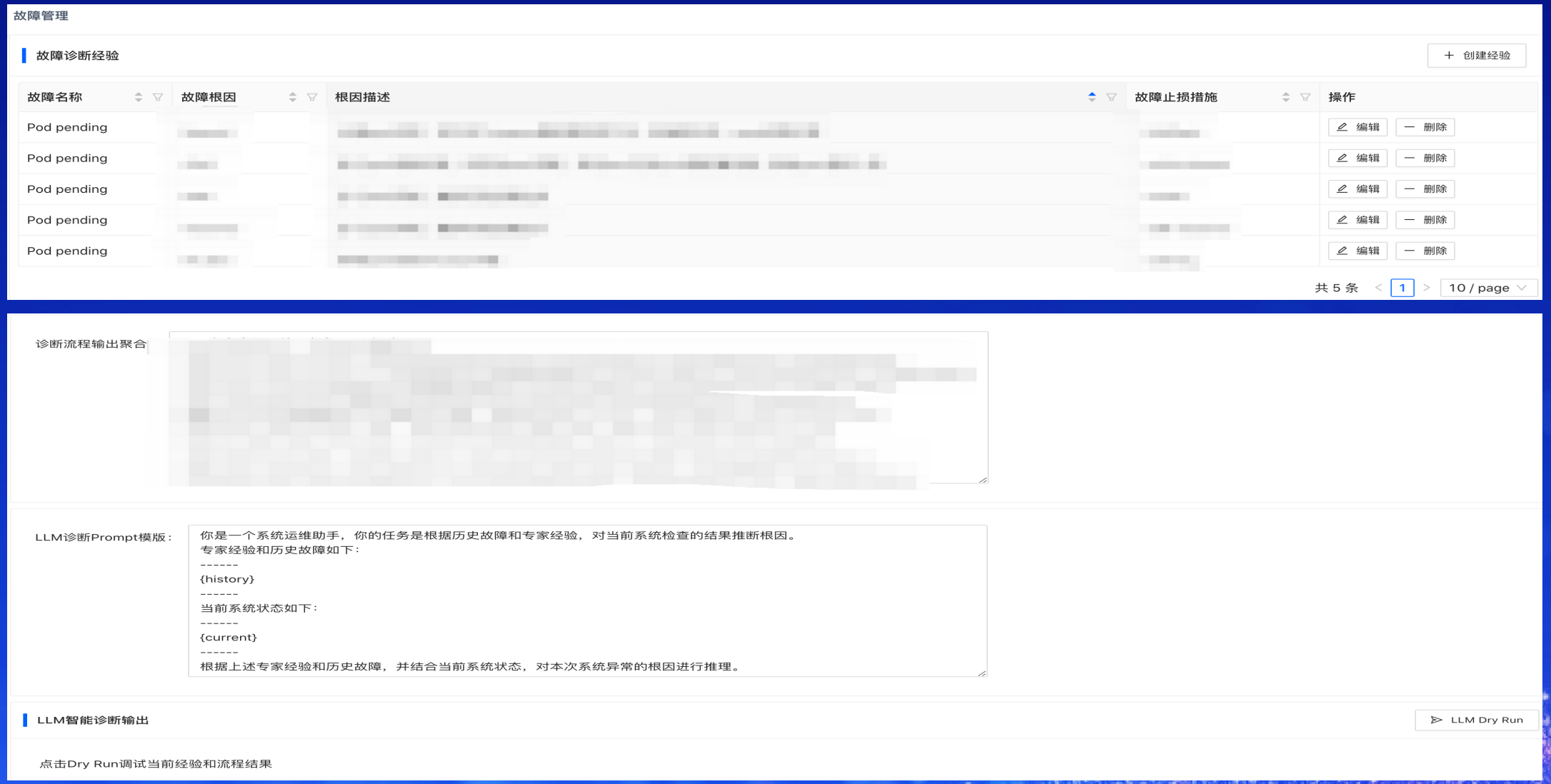
Agent的应用与调试

诊断项是否准确，以及输出是否符合预期，都可以由用户标注和修改



Agent的应用与调试

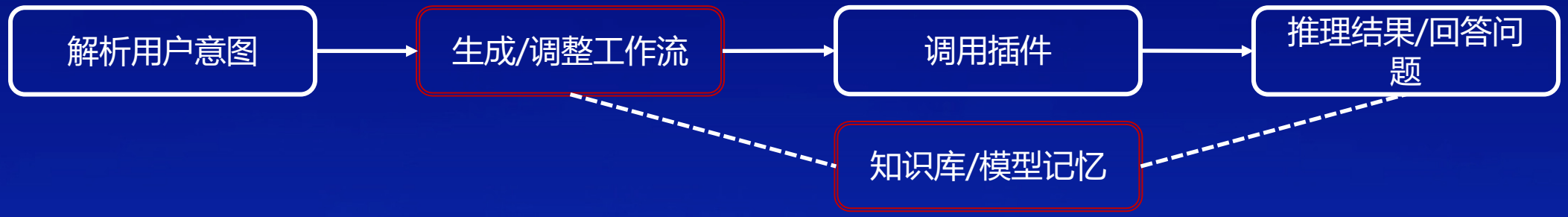
专家经验、诊断项输出、推理的Prompt，都支持由用户调试和修改



PART 05
一些探索中的经验教训

我们想做什么样的SRE-Copilot

LLM最吸引人的地方，在于可能会跨组件“通用”



1. 自动生成工作流?

组件SRE并不想配置固定工作流，尽管其可以准确的解决大部分问题

自动生成工作流的意义，在于可以让组件聚焦于高质量文档和个性化插件，而无需关心每一步该做什么。

2. 更灵活的Agent框架?

不是所有工作都能一步返回结果，Agent是在交互中完成任务的

动态根据系统反馈、人工确认和新信息输入进行决策，动态调整工作流，支持多人、多Agent协作。

3. RAG通过KG增强?

纯RAG只是基于检索，并不能保证所有相关信息都提供给LLM

如何通过知识图谱等形式更好更全的组织上下文，甚至是做一个Agent像人一样主动思考自己还不会什么，有针对性的去检索和学习。

4. 算法和业务如何合作?

LLM降低了用户的使用门槛，也降低了研发门槛和算法门槛

算法同学尽量把动态工作流、交互协作和知识库学习的通用性做好，和SRE一起动手，开发自己的Agent。

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

上海站
K+全球软件研发行业创新峰会
时间：2024.06.21-22

敦煌站
K+思考周®研习社
时间：2024.10.17-19

香港站
K+思考周®研习社
时间：2024.11.10-12

K+峰会详情

AiDD峰会

上海站
AI+研发数字峰会
时间：2024.05.17-18

北京站
AI+研发数字峰会
时间：2024.08.16-17

深圳站
AI+研发数字峰会
时间：2024.11.08-09

AiDD峰会详情

AiDD

THANKS