GOPS 2022
Shenzhen

GOPS 全球运维大会 2022

- XOps 风向标

深圳站

中国 · 深圳

指导单位：云计算开源产业联盟 FinOps产业推进方阵

主办单位：开放运维联盟 OOPSA Open OPS Alliance 高效运维社区 GreatOPS Community DevOps时代

时间：2022年4月21-22日

GOPS 2022
shenzhen

一站式测试平台

——低代码与云原生之路

王玺——虎牙

目录 Contents

/01

从测试造数开始

测试造数场景

功能测试前置条件：需要一个**用户账号**，关注了主播A，并且拥有5级粉丝牌

- 直播
  - 主播开播（RPC接口加密验证）
- 音视频
  - 持续推流（RTMP协议推流）
- 账号
  - 申请一个账号（三个关联的http接口）
- 用户
  - 订阅某主播（TARS-RPC接口）
- 支付
  - 充值金币（需签名校验的http接口）
- 礼物
  - 送礼提升亲密度（单次数额限制）
- 粉丝
  - 佩戴粉丝徽章

3 * 应用协议

5 * 关联的业务团队

7 * 步骤

9 * 接口请求

痛点分析

GOPS 2022 shenzhen

01

数据

1. 手工构造数据耗时占比**>20%**
2. 贴合业务场景的数据**构造难**

02

工具

1. 面对复杂业务流程，缺乏系统性抽象
2. **多样化的场景(协议)类型**传统接口工具不足以支撑

03

脚本

1. 对测试人员要求高
2. 脚本**可复用性差**
3. 脚本**可维护性差**

04

协作

1. 测试数据**重经验性**
2. 测试服务化？**开发+部署+运维**
3. 测试团队的**重复开发**

业界平台常见实现思路

Codeless

- 操作以UI交互为主
- 在接口、自动化工具上实现
- 可视化编排

Coding

- 操作以编码为主
- 统一的框架下进行编码
- 纯代码的逻辑编排

	优势	劣势
Codeless	1. 纯ui操作，易上手 2. 人员成本低	1. 复杂场景的新功能拓展不易 2. 复杂业务流程支持不佳
Coding	1. 灵活性极高，适配各种场景	1. 脚本维护成本高 2. 对QA人员要求高

/02

低代码与云原生之路

总体思路——造数流程拆解

GOPS 2022 shenzhen

原子性操作

- HTTP接口请求
- RPC接口调用
- 数据库查询
- 数据库插入
- 随机生成
- 数据包截取、解析

……

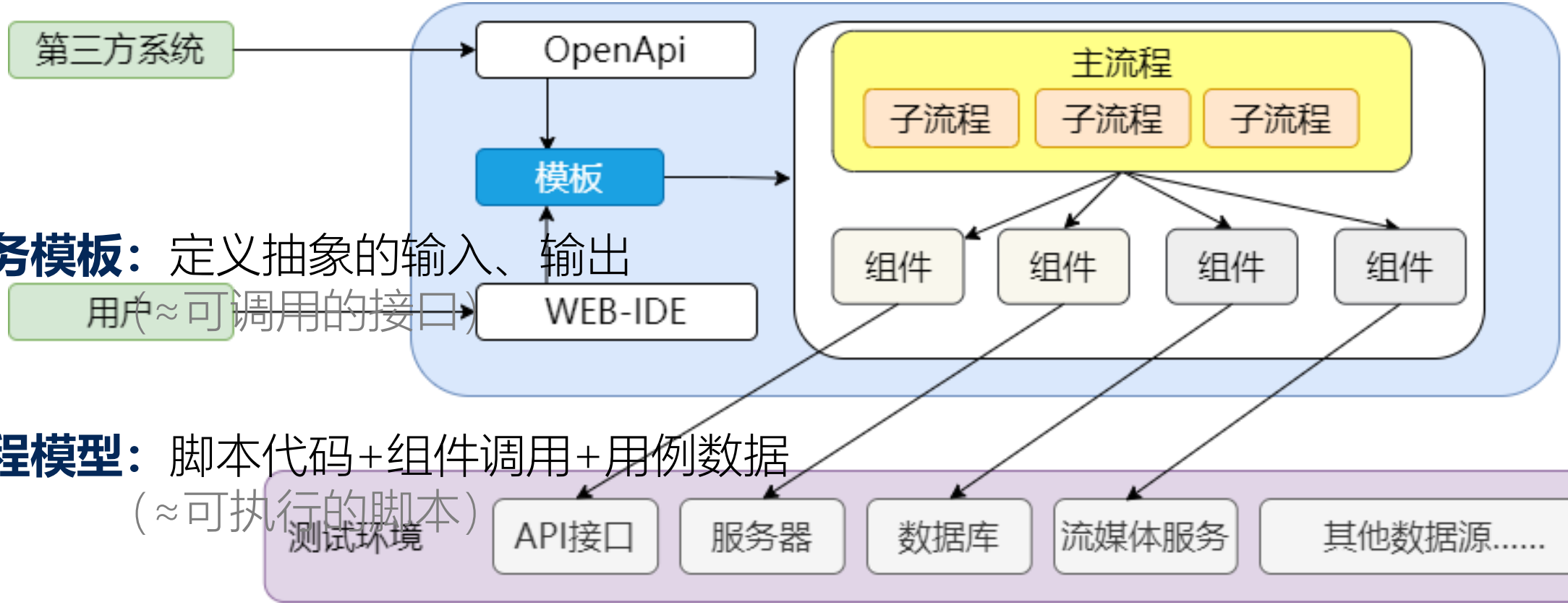
流程编排

- 并行、串行
- 循环调用
- 选择判断
- 延时
- 异常跳过
- ……

易用性封装

- 提取入参、出参
- 枚举参数（单选、多选）
- 参数默认值、参数校验
- 批量执行
- 界面调用 & 接口调用
- ……

总体思路——调用流程

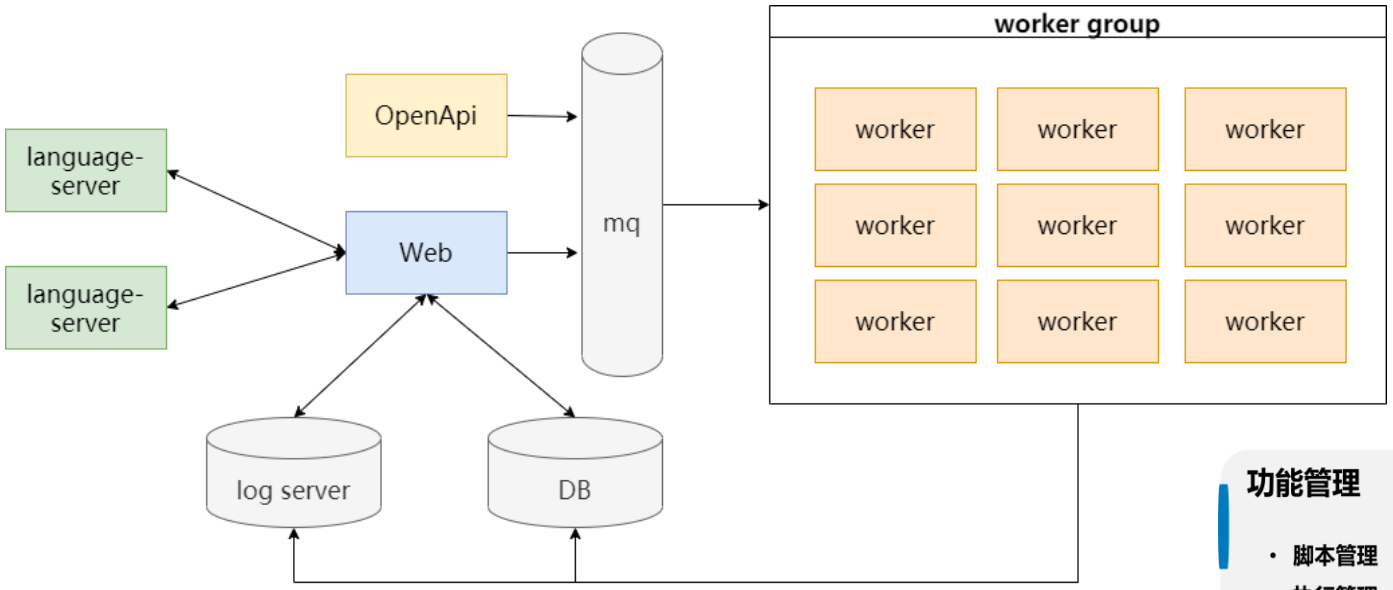


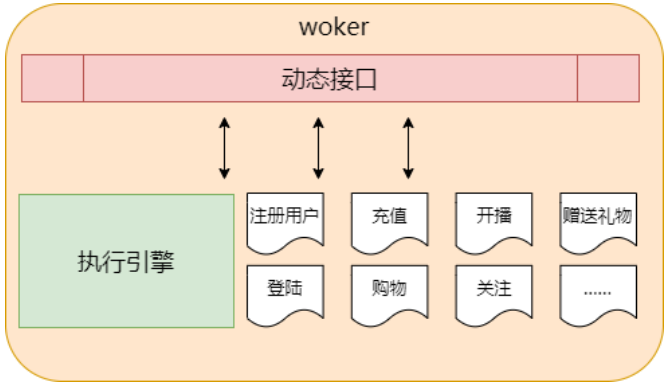
业务模板：定义抽象的输入、输出
（≈可调用的接口）

流程模型：脚本代码+组件调用+用例数据
（≈可执行的脚本）

功能组件：复杂功能的封装，供流程调用
（≈功能函数）

总体思路——早期测试引擎设计





Web：前端，负责脚本创建、编辑等工作

Language-server：为WEB-IDE提供语法分析等服务

Worker：任务执行节点，无状态水平扩容

OpenAPI：对外提供接口

功能管理

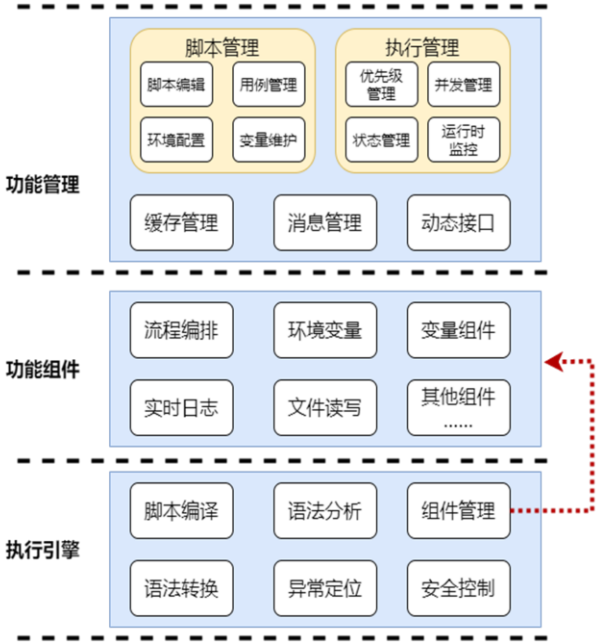
- 脚本管理
- 执行管理

功能组件

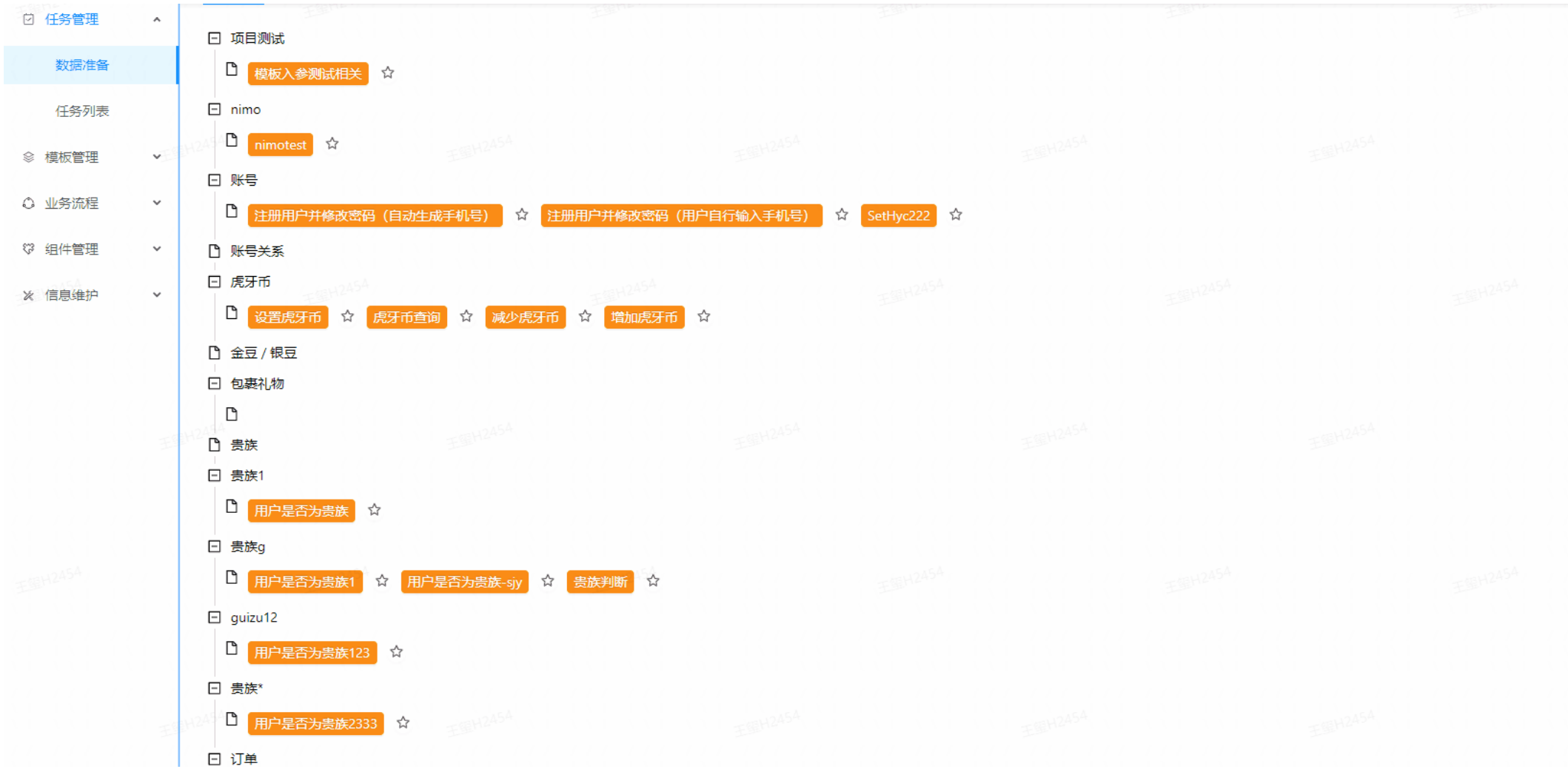
- 系统核心组件
- 其他功能组件

执行引擎

- 脚本引擎
- 组件系统



使用演示



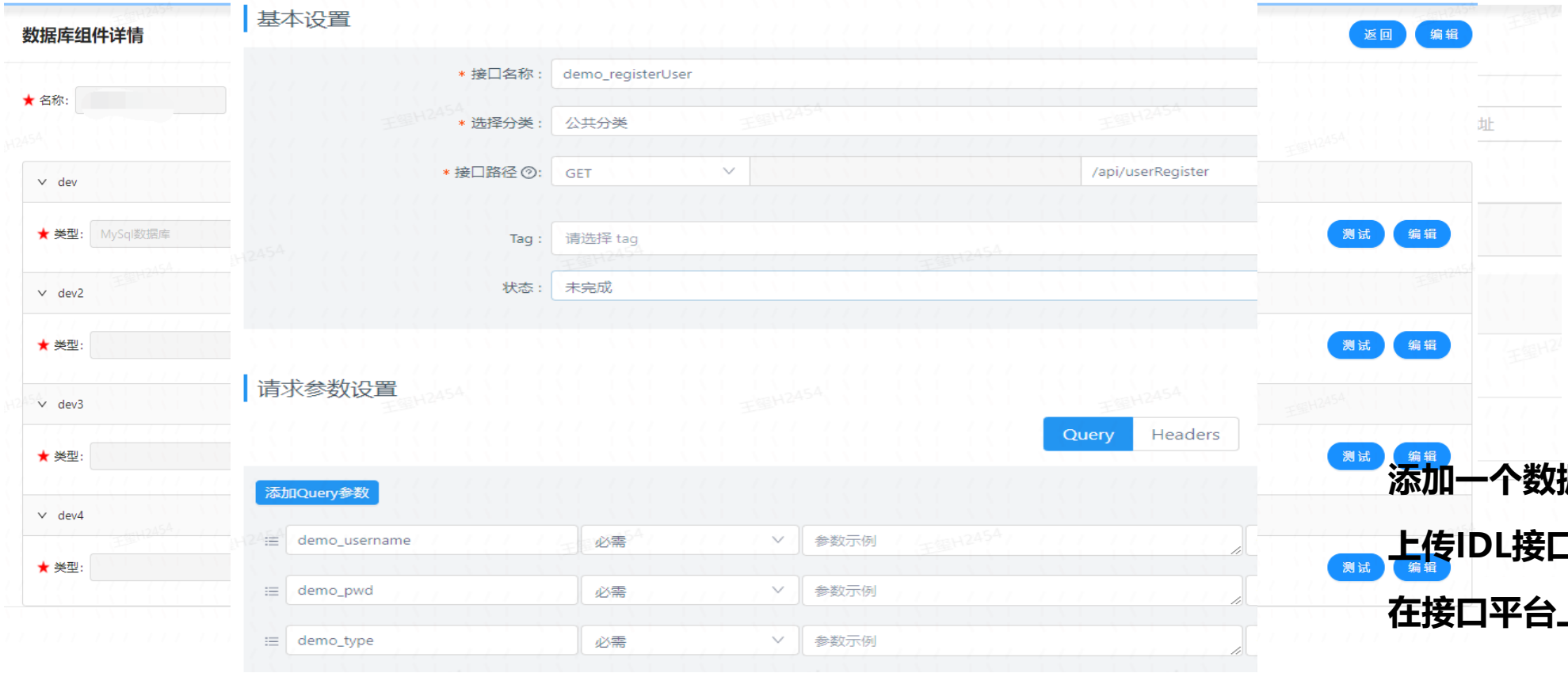
注：动图内容仅为演示时创建的测试数据

使用演示——组件配置录入

录入HTTP接口

录入RPC接口

录入数据库配置



添加一个数据库配置

上传IDL接口文件

在接口平台上录入

使用演示——流程编写，用例录入

编写流程脚本　　录入流程用例　　调试执行


```
pf.http.requestForObj("user","userRegister")
--------------
pf含义为public function，即组件方法
```

声明式调用组件

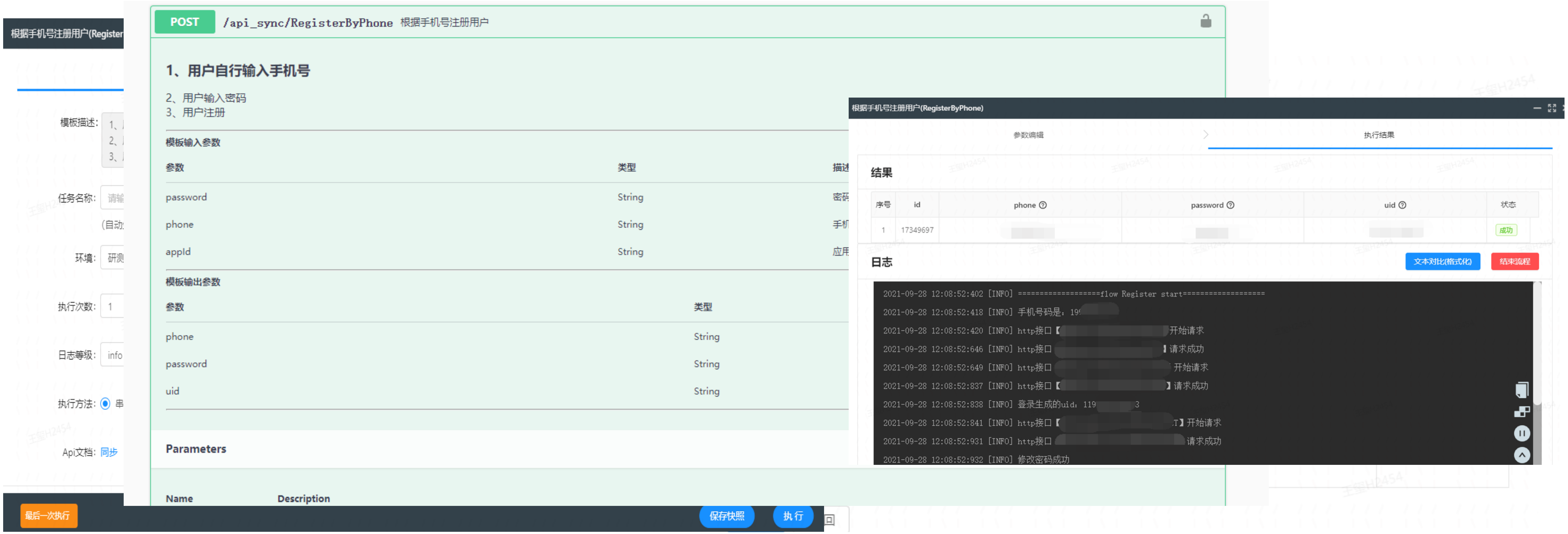
口参数（表头），按需填写用例

即时编译，实时日志

使用演示——模板提取并使用

GOPS 2022
shenzhen

提取输入输出参数　　界面手动调用　　接口调用



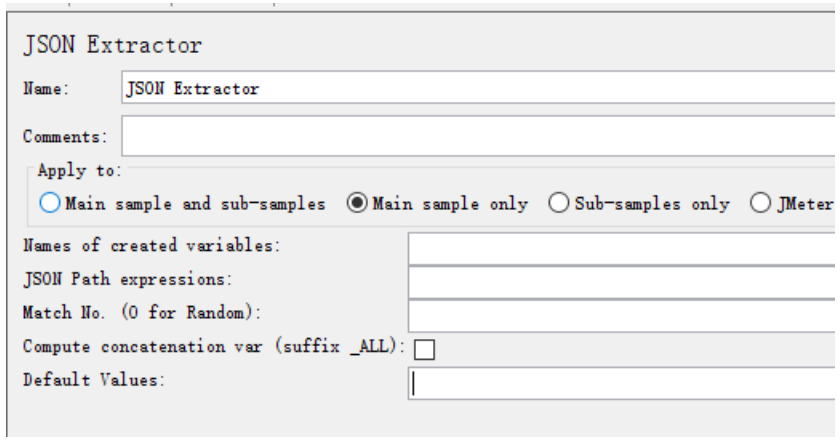
不停机，热更新，动态生成接口与文档

系统从脚本中解析参数，按需提……默认值、校验

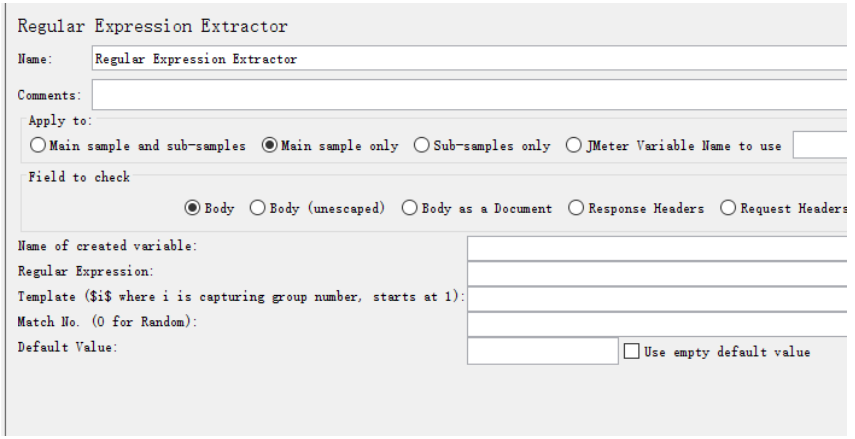
低代码组件——使用方式（配置+调用）

GOPS 2022 shenzhen

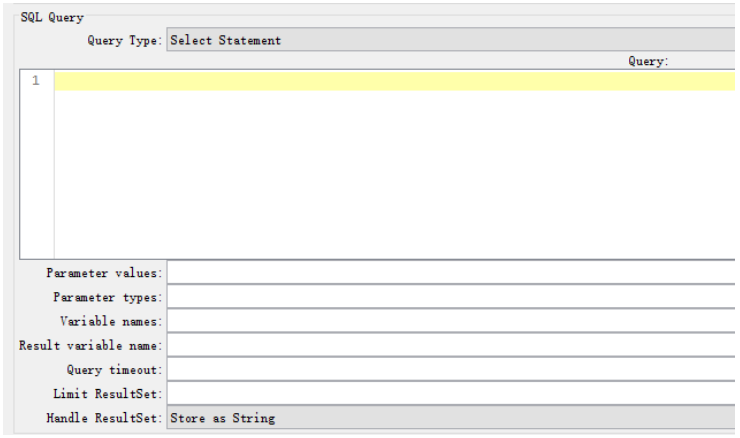
Json提取



正则匹配



SQL查询




```
def resp = pf.http.requestForString("user-service", "re
def title= pf.json.path(resp,"$.store.book[0].title")//
pf.regex.matchAll(title,"(?<=body).+(?=hu)")//regex
pf.mysql.execute("db1","select * from xxx")//sql
```

```
// }

@Override
void test() {

}

//  @Override
//  void after() {
```


huya

低代码组件——生成式数据表驱动

传统编写一个http请求需要传入哪些参数?

```
axios({
    method: 'post',
    url: '/user/12345',
    data: {
        firstName: 'Fred',
        lastName: 'Flintstone'
    },
    Headers:{
        "agent":"xxxx"
    },
    Proxy:{
        host: '127.0.0.1',
        port: 9000
    }
});
```

header url query path body proxy……

弊端：

脚本内代码繁琐，不易理解

多套测试环境，增加代码复杂度

多人协作，脚本难以复用

低代码组件——生成式数据表驱动

GOPS 2022 shenzhen

核心理念：命令式编程——》声明式编程

```
pf.http.requestForObj("user","userRegister")
```

+

	demo_username	demo_pwd	demo_type
注册普通用户	Fred	Flintstone	xxxx
注册VIP用户	Foo	Bar	xxxx
注册超级管理员	Fred	Flintstone	xxxx

自动生成参数

★ 流程名称：PPTDemo　流程编辑　流程用例　表头筛选　日志等级：debug　环境：dev　表格内搜索

	A	B	C	D	E	F	H	I	J	K
1				流程用例	用例描述	username	userType	HTTP：user-service->apiUserRegisterGET		
2								demo_username	demo_pwd	demo_type
3	运行	保存	×	PPTDemo_common_user	注册一个普通用户	user-090931	0	${username}	${pwd}	${userType}
4	运行	保存	×	PPTDemo_vip_user	注册一个vip用户	user-090932	1	${username}	${pwd}	${userType}
5	运行	保存	×	PPTDemo_black_user	注册一个黑名单用户	user-090933	2	${username}	${pwd}	${userType}
6	运行	保存	×							

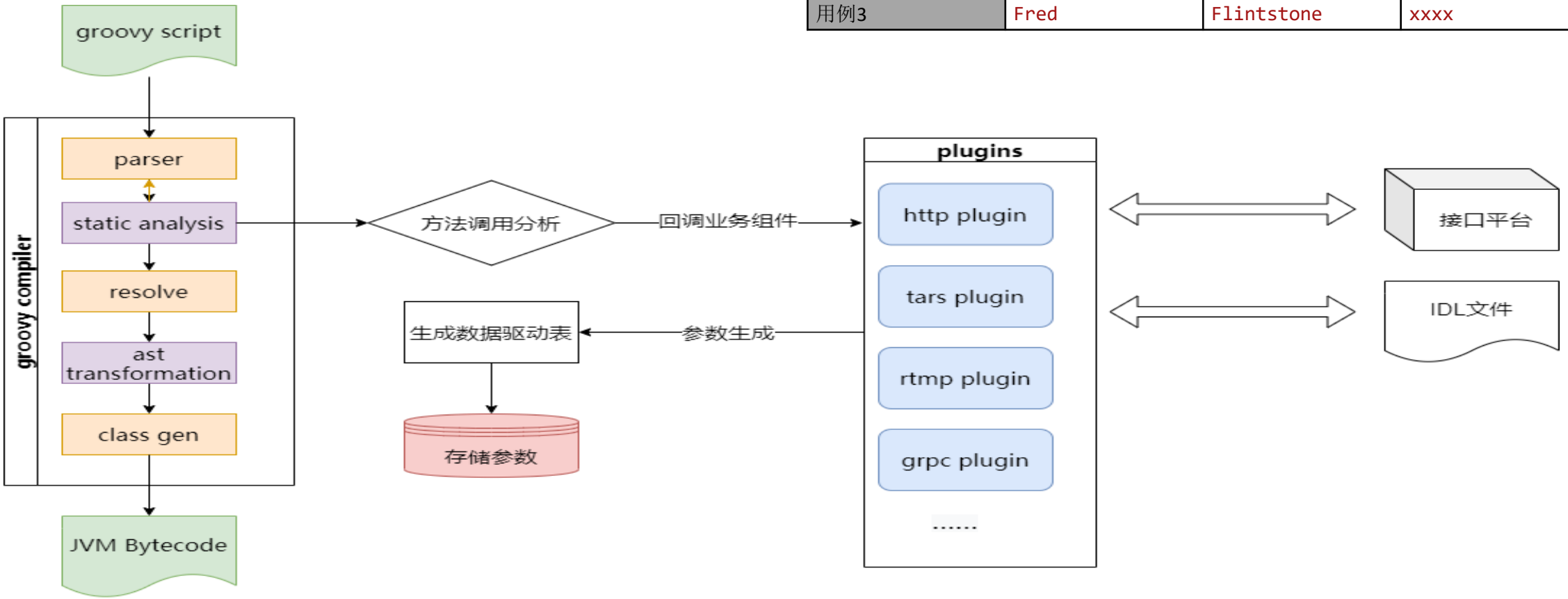
huya

低代码组件——编译期的回调设计

如何从声明式的调用分析出请求参数?

```
pf.http.requestForObj("user","userRegister")
```

	demo_username	demo_upwd	demo_type
用例1	Fred	Flintstone	xxxx
用例2	Foo	Bar	xxxx
用例3	Fred	Flintstone	xxxx

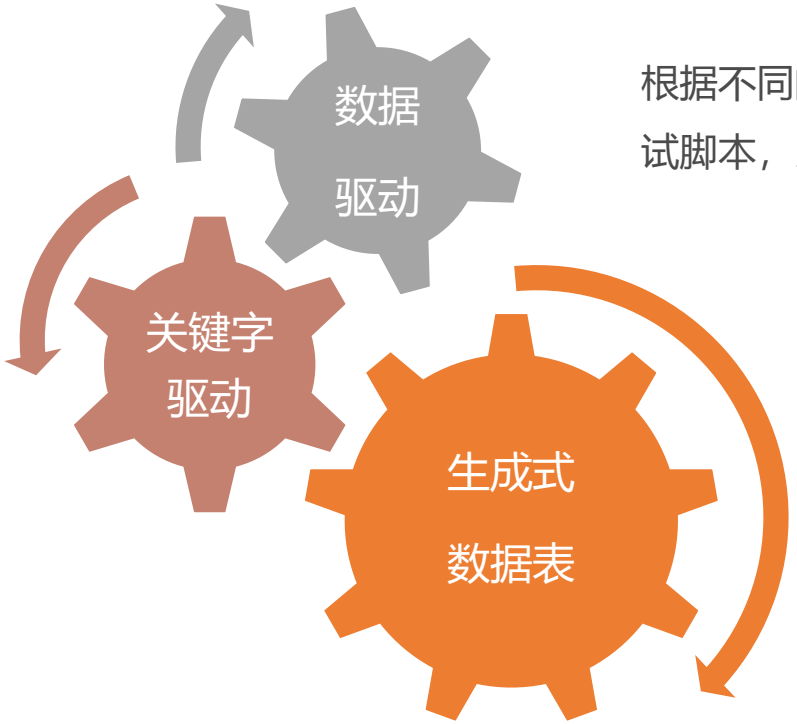


低代码组件——从数据驱动到关键字驱动


```
pf.http.requestForObj("user","userRegister")
```

+

	demo_username	demo_pwd	demo_type
注册普通用户	Fred	Flintstone	xxxx
注册VIP用户	Foo	Bar	xxxx
注册超级管理员	Fred	Flintstone	xxxx



根据不同的测试环境，测试用例，运行同一测试脚本，即**测试逻辑与测试数据的分离**

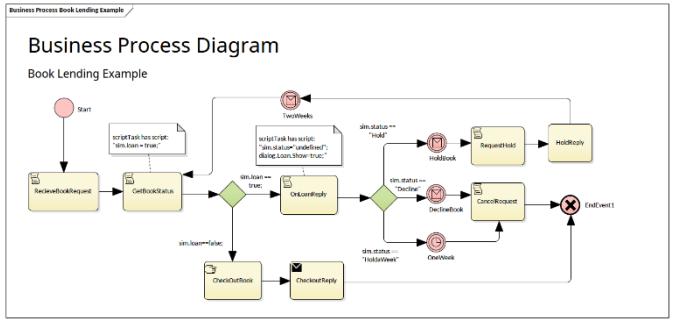
在数据驱动的基础上，通过**关键行为（关键字）拓展测试用例的模式**，从而达到一个由数据和关键字驱动整个测试的效果

由系统自动完成参数化。测试同学通过声明式的编程，从而**更专注于当前测试逻辑的完成**

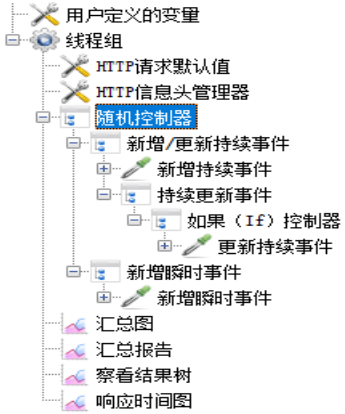
低代码编排——常见的一些流程编排模式

GOPS 2022 shenzhen

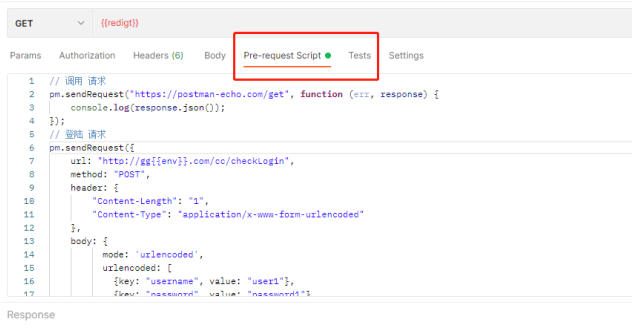
工作流引擎界面拖拽式编排



Jmeter控制器编排



Postman接口前后置编排



HttpRunner钩子函数式编排

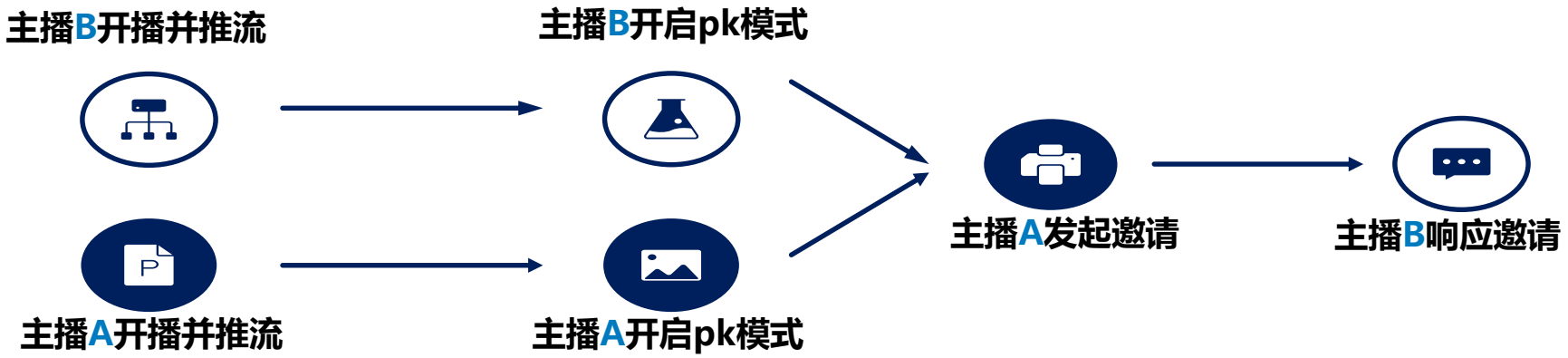
```
- config:
    name: basic test with httpbin
    request:
        base_url: http://127.0.0.1:3458/
    setup_hooks:
        - ${hook_print(setup_testset)}
    teardown_hooks:
        - ${hook_print(teardown_testset)}

- test:
    name: get headers
    times: 2
    request:
```


huya

编排——低代码流程编排带来的改变

GOPS 2022 shenzhen

传统流程编排样例



- 复杂流程编排缺乏灵活、拓展
- 执行过程黑盒，难以debug
- 新功能的开发成本高
- 对工作流界面的适应学习成本

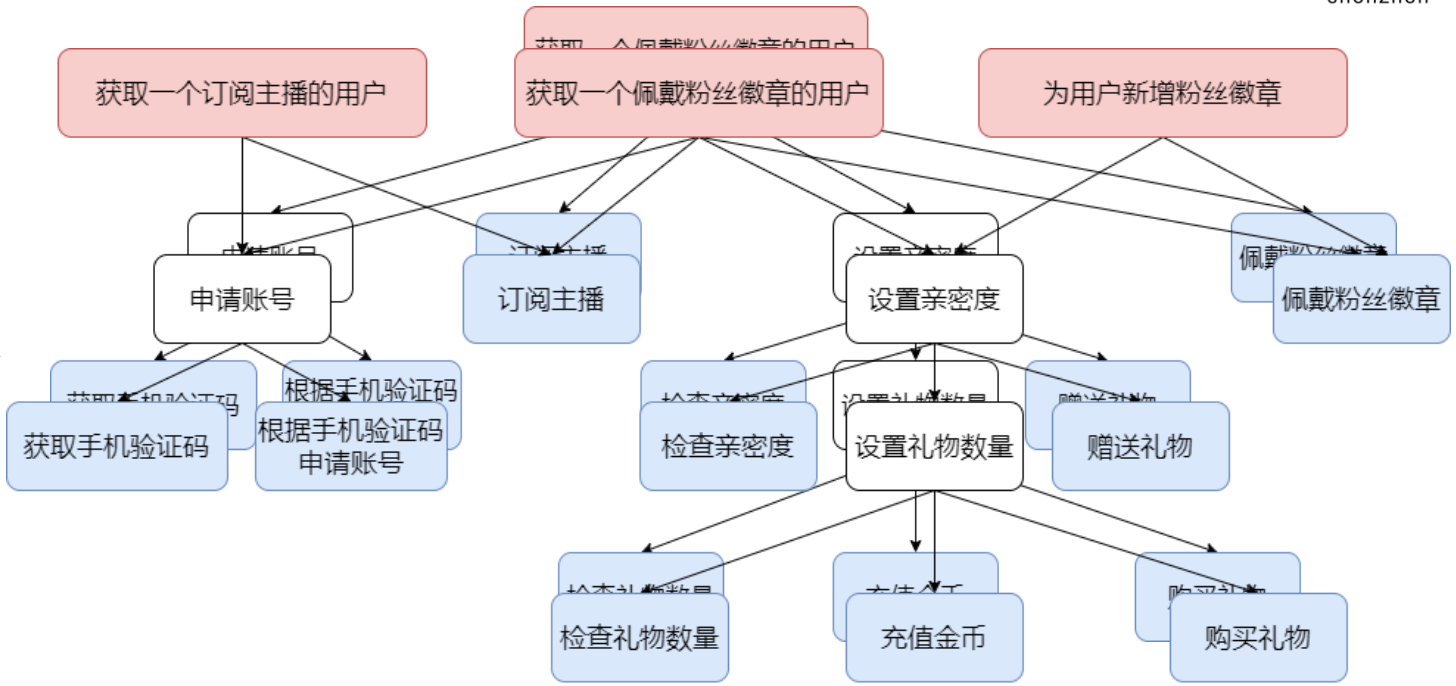
低代码编排样例

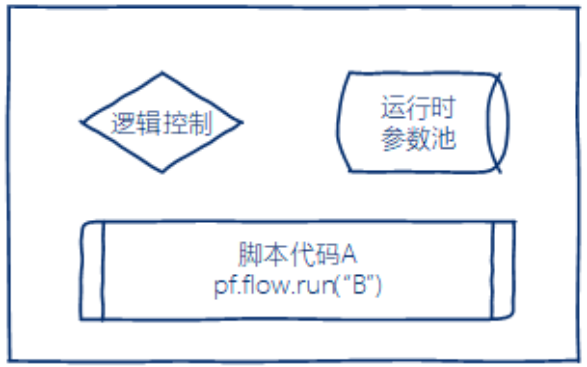
```
pf.flow.parallel(
    {
        pf.flow.runByAlias("BeginLiveAndPushStream","BeginLive_A")
        pf.flow.runByAlias("StartPKMode","Start_A") },
    {
        pf.flow.runByAlias("BeginLiveAndPushStream","BeginLive_B")
        pf.flow.runByAlias("StartPKMode","Start_B") })
if("${Start_A.status}"=="${Start_B.status}==0"){
    pf.flow.run("InviteMultiPK") //邀请
    pf.flow.run("ResponseMultiPKInvite") //接收
}
```

- 原生代码编排，简单易上手
- 代码执行，可以打日志or调试
- 更低的组件开发成本
- 会写代码就能上手

低代码编排——标准流程模型相关

- 账号：申请一个账号
- 用户：订阅某主播
- 支付：充值金币
- 礼物：送礼提升亲密度
- 粉丝：佩戴粉丝徽章





复杂流程由**基础流程**组合而成，

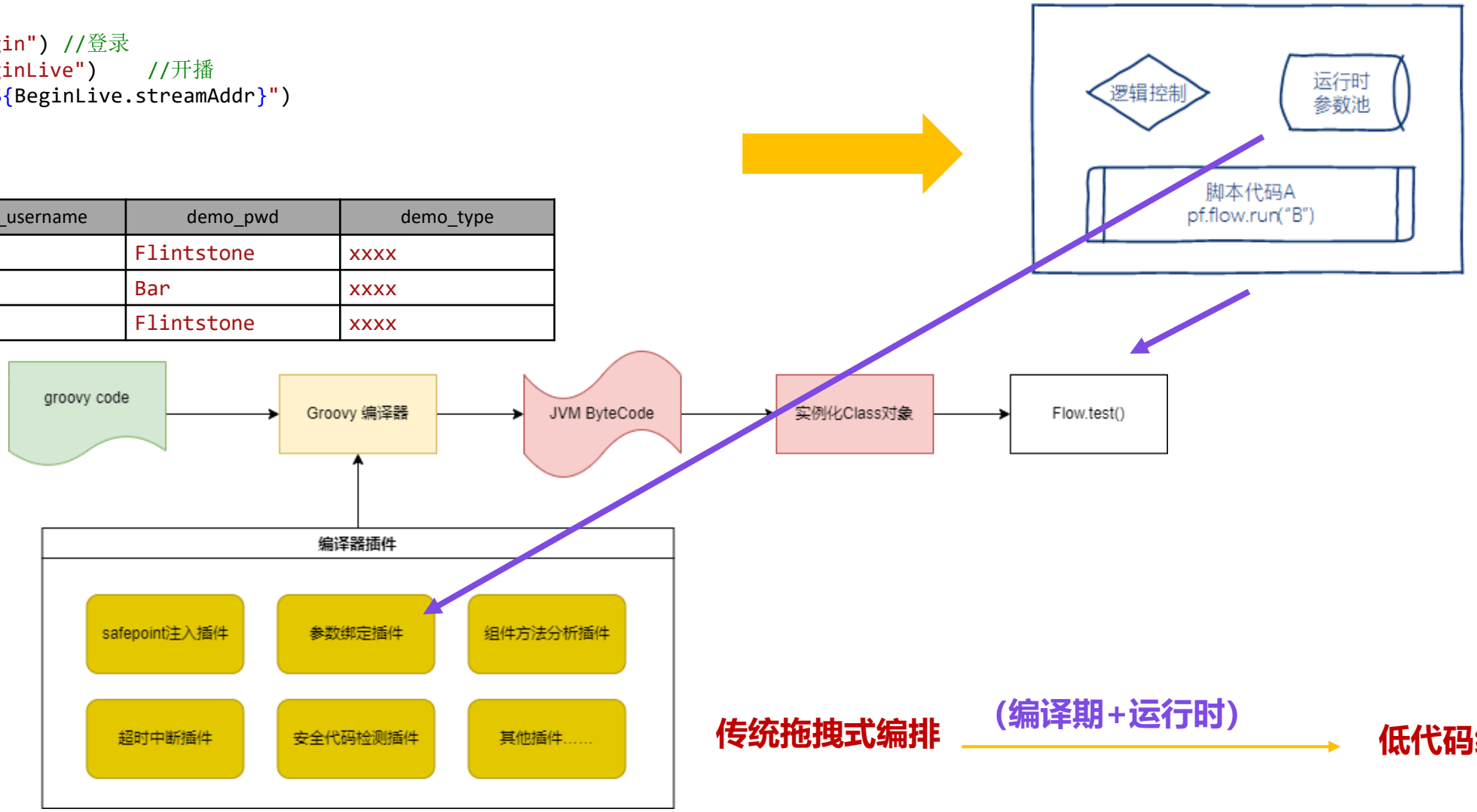
流程间相互**复用**，打造贴合业务的流程**生态库**

低代码编排——标准流程模型对业务逻辑的封装

GOPS 2022 shenzhen

```
class BeginLiveAndPushStream extends Flow {
    @Override
    void test() {
        pf.http.run("Login") //登录
        pf.flow.run("BeginLive")     //开播
        pf.rtmp.push ("${BeginLive.streamAddr}")
        //调用推流组件
    }
}
```

	demo_username	demo_pwd	demo_type
普通用户开播	Fred	Flintstone	xxxx
VIP用户开播	Foo	Bar	xxxx
黑名单用户开播	Fred	Flintstone	xxxx



传统拖拽式编排 (编译期+运行时) 低代码编排

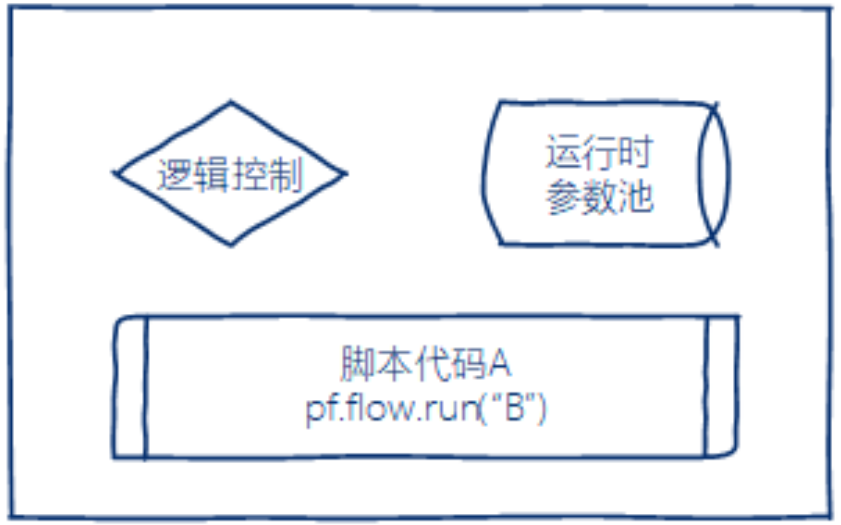
huya

低代码编排——更深度的流程控制能力

GOPS 2022

代码脚本在多人协作遇到的痛点

- **逻辑与数据耦合**：多场景下难以复用，参数化的习惯难以养成
- **统一的编排**：个人编码习惯不同，代码风格迥异，难以进行




```
pf.flow.run("BeginLive")
def streamAddr = pf.variable.get("BeginLive", "streamAddr")
//调用推流组件
if(streamAddr){
    pf.rtmp.pushImage(streamAddr)
}
for (def uid in [123,456,788,960]){
    pf.flow.runWithVar("BeginLive",["phone": uid])
}//循环给多人开播
```

①代码**逻辑**与**数据**的分离

	BeginLive	phone	pwd
用例1	用户A开播	1881959	123
用例2	用户B开播	1881958	456

③**无侵入**地修改**子流程**参数

②静态分析代码后自动完成**参数化**

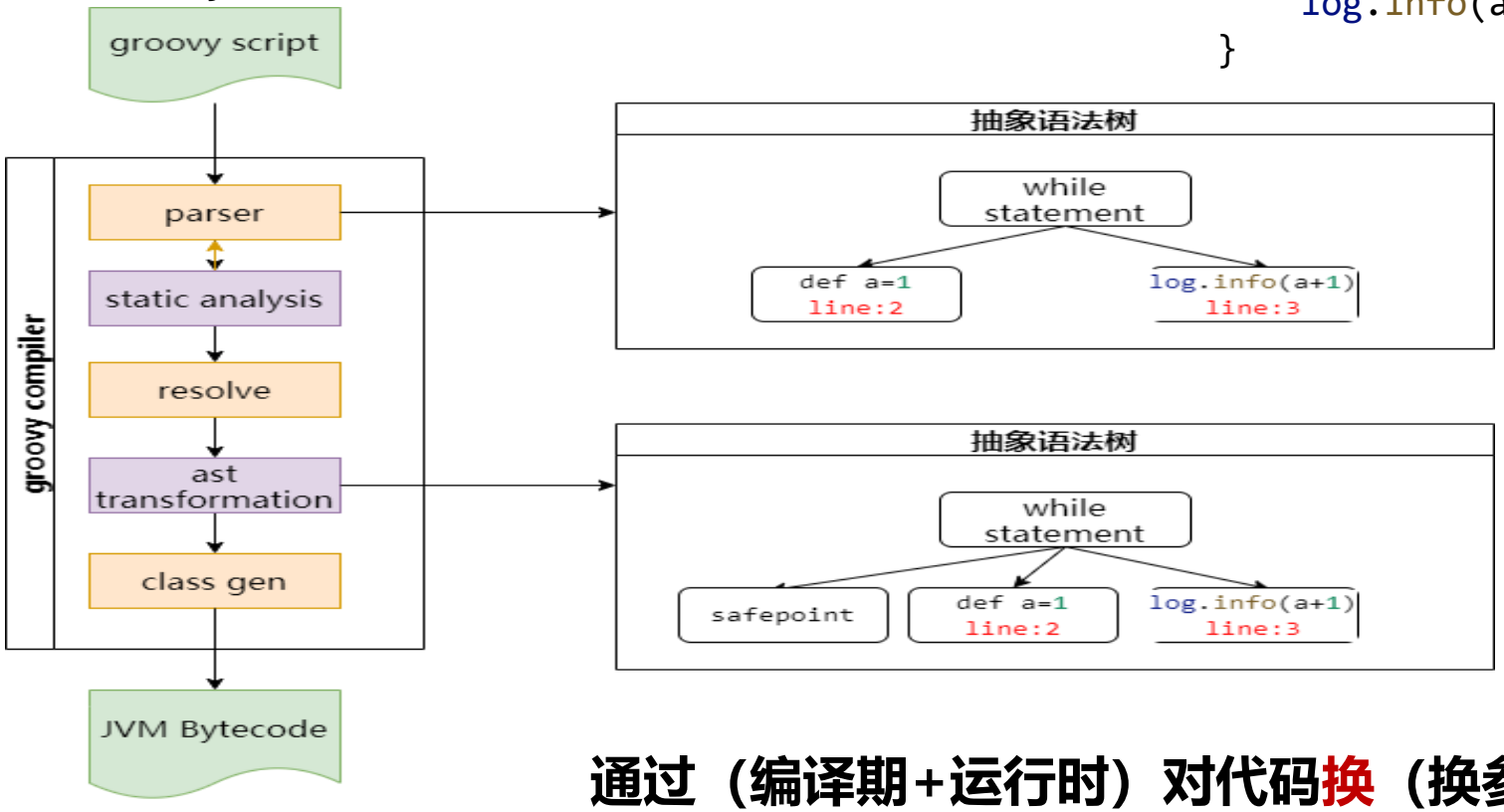
huya

低代码编排——“编译期”OR“运行时”代码注入

GOPS 2022 shenzhen

```
while(true){
    def a=1
    log.info(a+1)
}
```

```
while (true) {
    if (Condition.shouldStop(Thread.currentThread())) {
        Thread.currentThread().wait();
    }
    def a=1
    log.info(a+1)
}
```



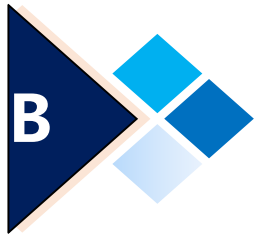
- 稳定性需求
- 安全性保障
- 参数可替换
- 逻辑可修改
- 流程可编排

通过（编译期+运行时）对代码换（换参数）、排（编排流程）、改（改逻辑）

低代码编排——低代码编排小结

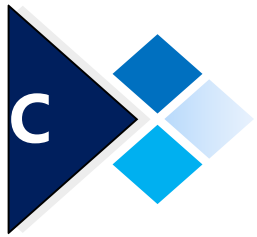
更灵活的编排方式

使用原生代码，自由度极高。
例如失败重试，循环，判断等



无侵入式流程控制

父流程可以在不改动子流程情况下完成对运行参数，运行逻辑的控制



封装实现细节

声明式调用，隐藏实现细节。
测试只需要专注测试逻辑即可

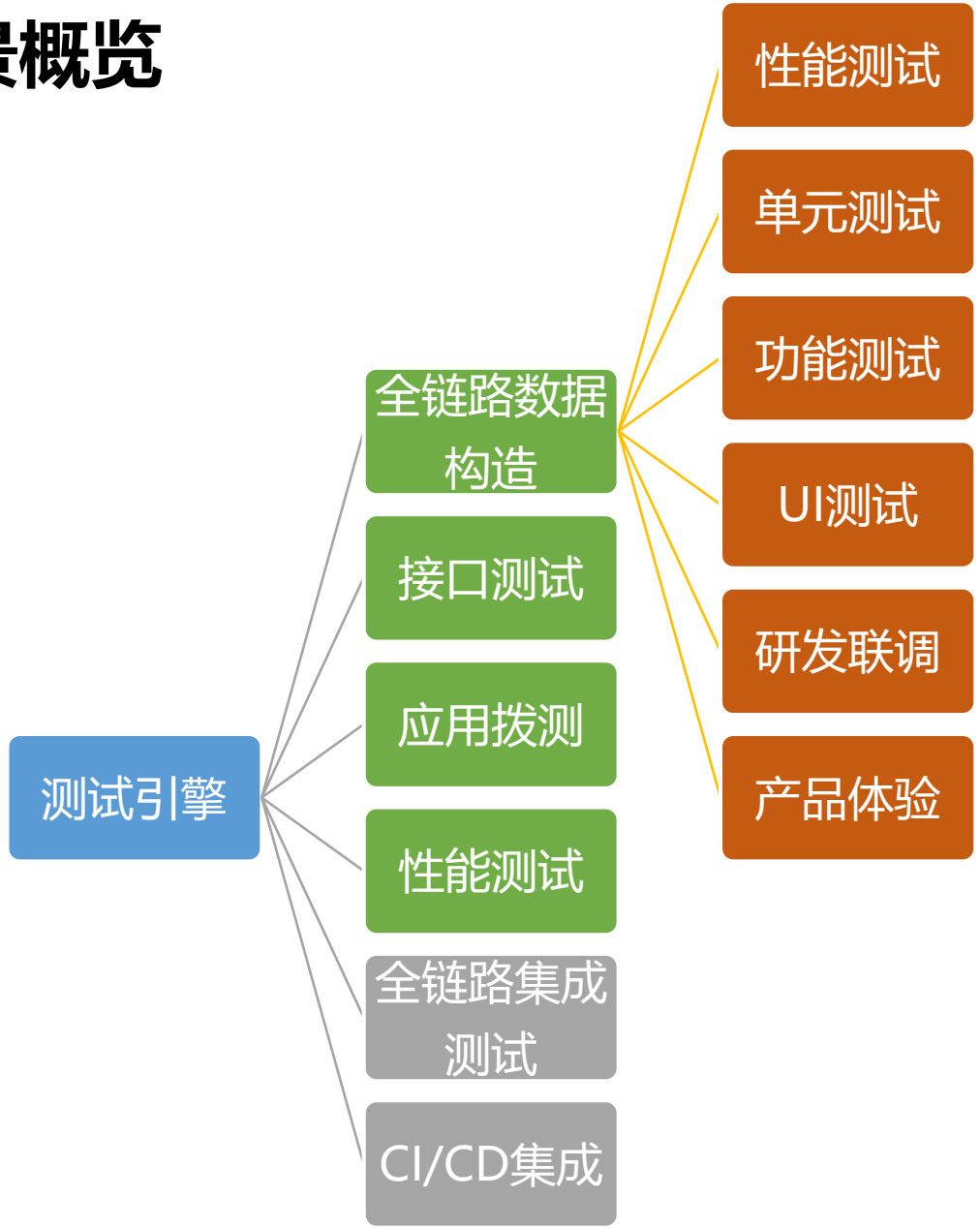
统一的流程模型

系统层面：方便后续功能拓展
脚本层面：可维护行极大提升

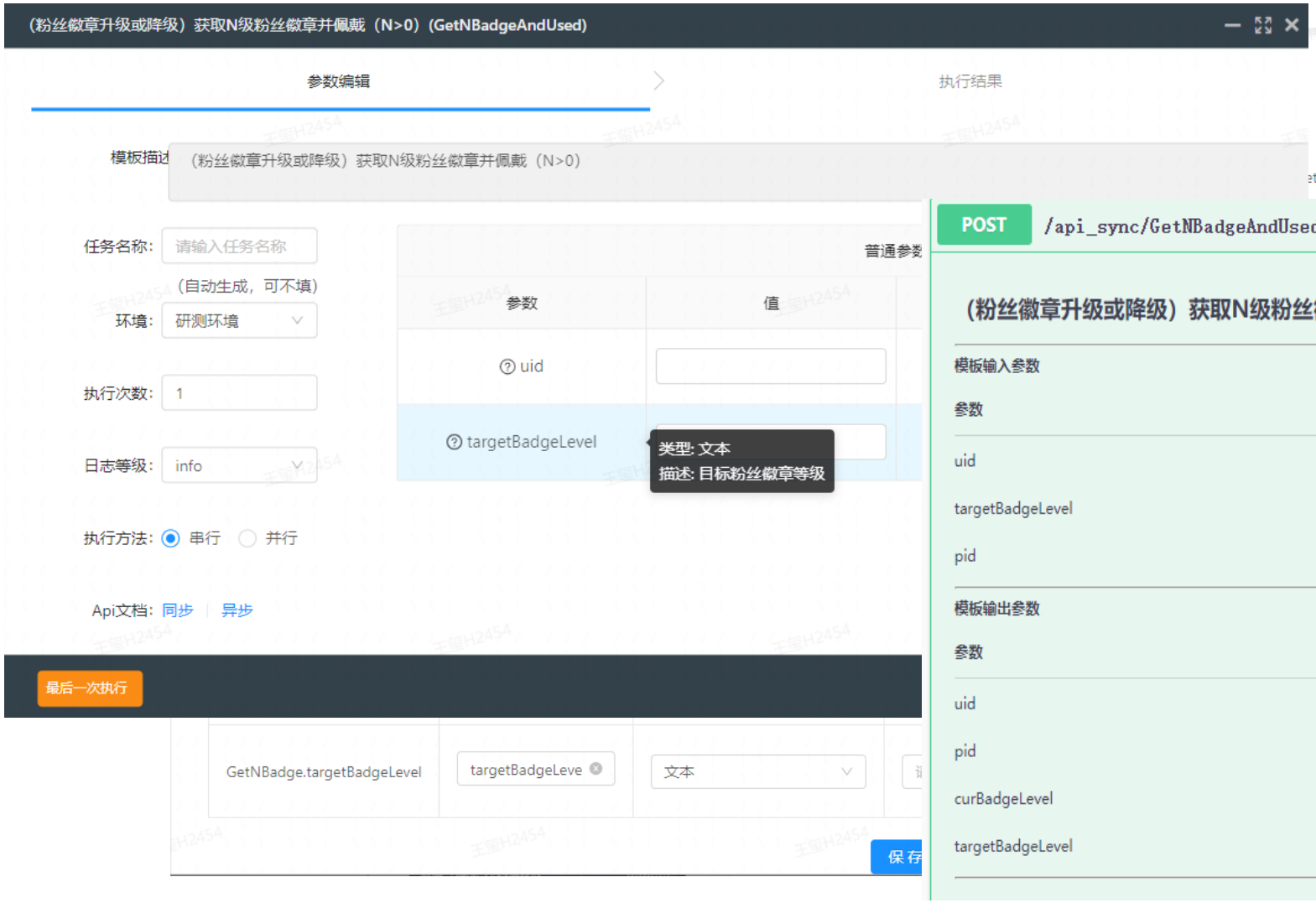
/03

一站式探索

一站式应用场景概览


测试引擎
全链路数据构造
接口测试
应用拨测
性能测试
全链路集成测试
CI/CD集成
性能测试
单元测试
功能测试
UI测试
研发联调
产品体验

复杂场景造数——获取N级粉丝徽章



POST /api_sync/GetNBadgeAndUsed （粉丝徽章升级或降级）获取N级粉丝徽章并佩戴（N>0）

（粉丝徽章升级或降级）获取N级粉丝徽章并佩戴（N>0）

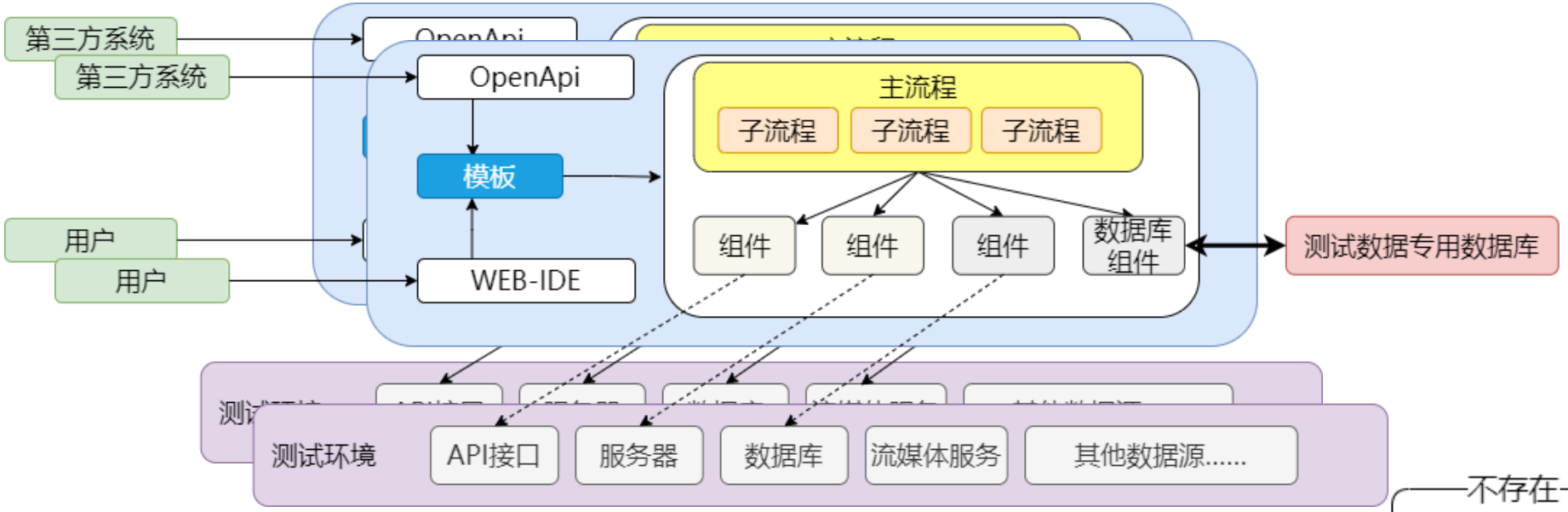
模板输入参数

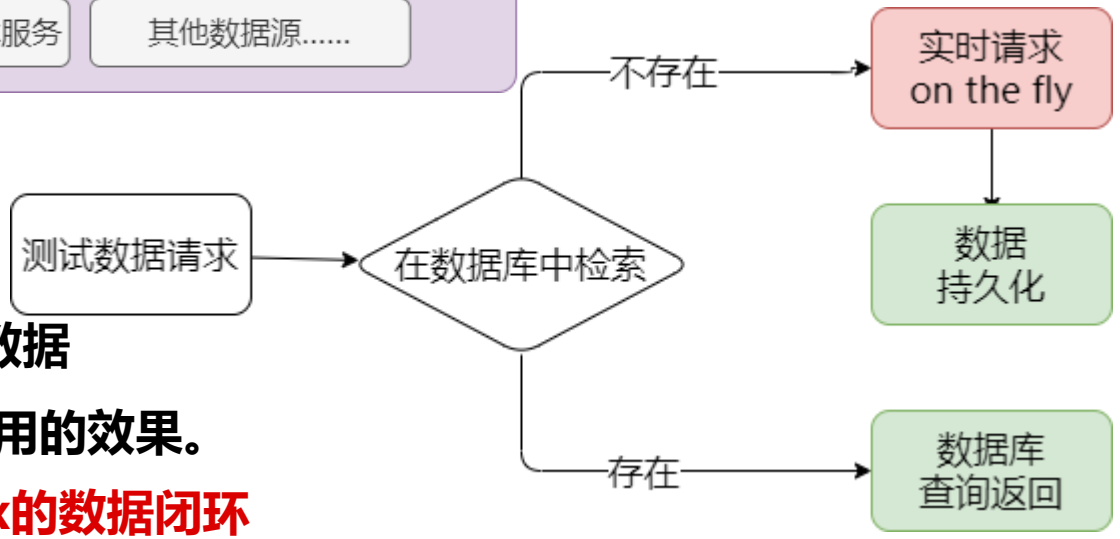
参数	类型	描述
uid	String	用户uid
targetBadgeLevel	String	目标粉丝徽章等级
pid	String	主播uid

模板输出参数

参数	类型	描述
uid	String	用户uid
pid	String	主播uid（徽章id）
curBadgeLevel	String	当前粉丝徽章等级
targetBadgeLevel	String	目标粉丝徽章等级

造数从On-the-fly到Out-of-box





on-the-fly：执行用例时，再根据用例需要创建测试数据

out-of-box：提前创建测试数据，提测后达到开箱即用的效果。

通过数据库组件低成本实现On-the-fly到Out-of-box的数据闭环

接口自动化——用户登录接口

GOPS 2022 shenzhen

①接口调用+断言

★流程名称：LoginServDemo 流程编辑 流程用例

```
1  package flow
2  // flow desc:
3  class LoginServDemo extends Flow {
4
5      @Override
6      void before() {
7          pf.variable.declare("user")
8      }
9
10     @Override
11     void test() {
12         def resp=pf.http.requestForObj("LoginServ","loginPOST")
13         assert resp?.ret==0
14     }
```

②数据表驱动，编写用例

★流程名称：LoginServDemo 流程编辑 流程用例 表头筛选 日志等级：debug 环境：dev

表格内搜索

	A	B	C	D	E	F	G
1				流程用例	用例描述	user	HTTP: LoginServ->login POST
2							root
3	运行	保存	×	LoginServDemo_comm	普通用户登录	user_comm	"pwd": "123456",
4	运行	保存	×	LoginServDemo_vip	VIP用户登录	user_vip	"pwd": "123456",
5	运行	保存	×	LoginServDemo_black	黑名单用户登录	user_black	"pwd": "123456",
6	运行	保存	×				

root

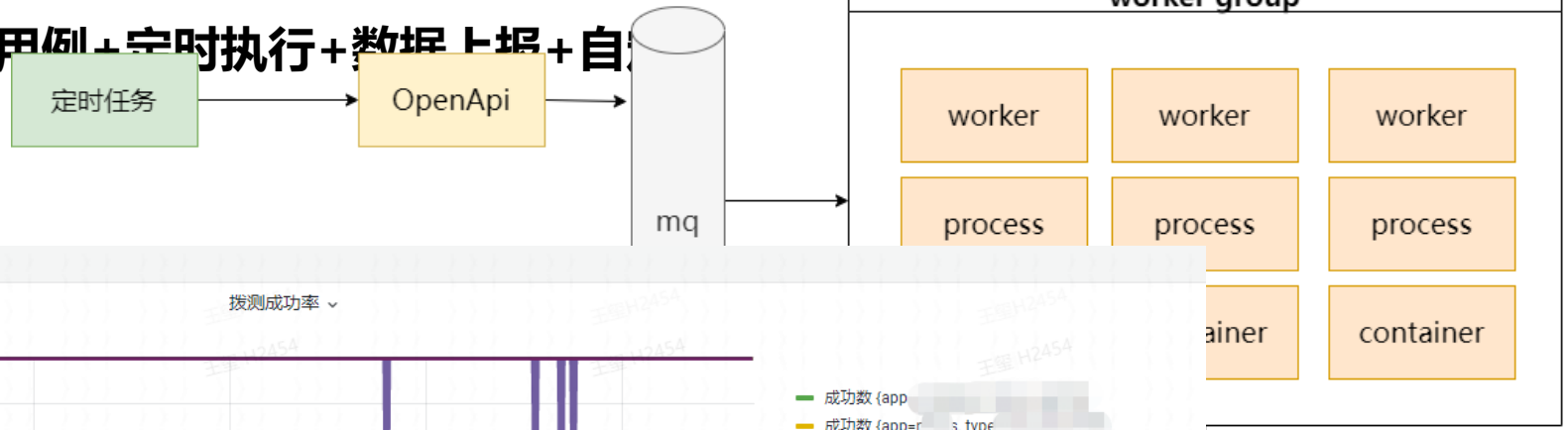
```
{
  "user": "${user}",
  "pwd": "123456",
  "app": 17891
}
```

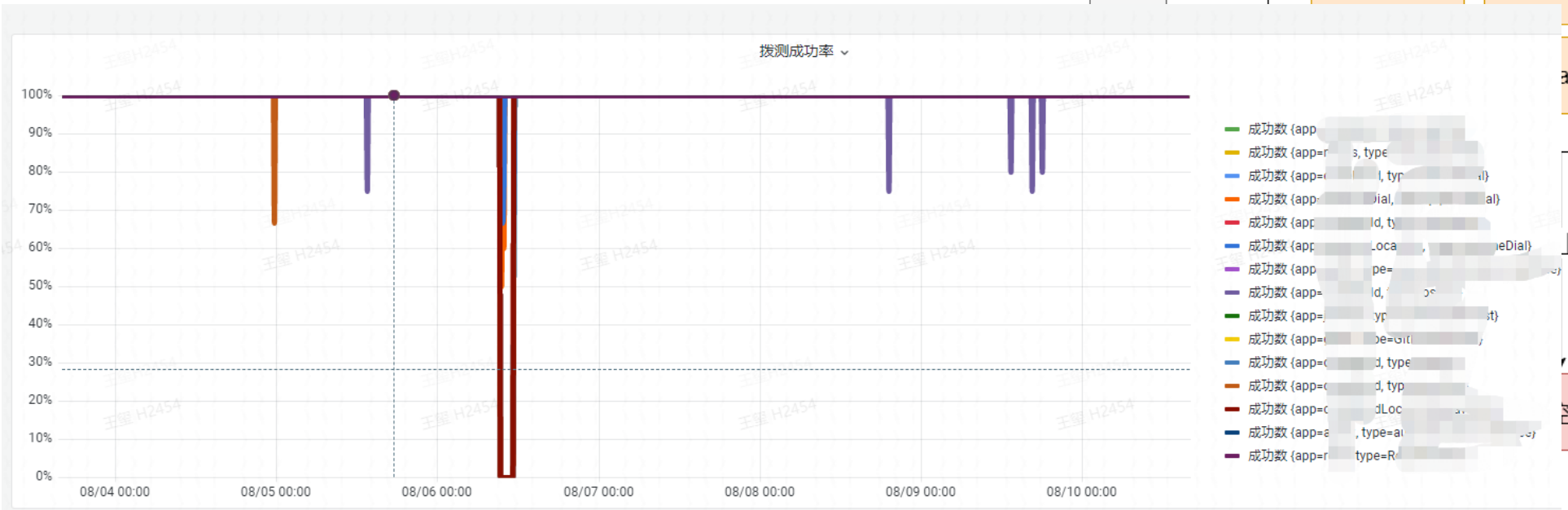
声明接口 → 编写断言 → 录入用例 → 自动化任务

huya

业务级应用拨测

1. 案例一：造数脚本的稳定性监控——反馈测试环境应用稳定性
2. 案例二：流程脚本+用例+定时执行+数据上报+自动监控告警





上报

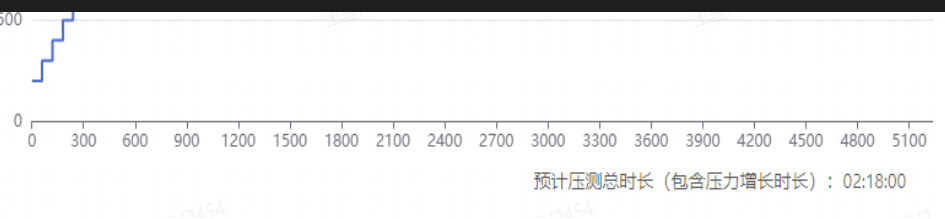
+告警

性能测试

GOPS 2022
shenzhen

未专项开展前，测试同学自主实现的一个简单的压测脚本

```
void tes
    def t    @Override
    def t    void test() {
    //自        def resp=pf.http.requestForObj("LoginServ","loginPOST")
    Execu        assert resp?.ret==0                                    m, new ThreadFactoryBuilder().setNameFormat("l                    ").build());
    for (    }
        executorService.execute(new Runnable() {
            void run() {
                try {
                    pf.taf.requestForObj("HiCompare", "JMG.H                    _V2")
                } catch (Exception e) {
                    log.error("", e)
                }
            }
        })
    }
    executorService.shutdown();
    executorService.awaitTermination(10, TimeUnit.HOURS)
```

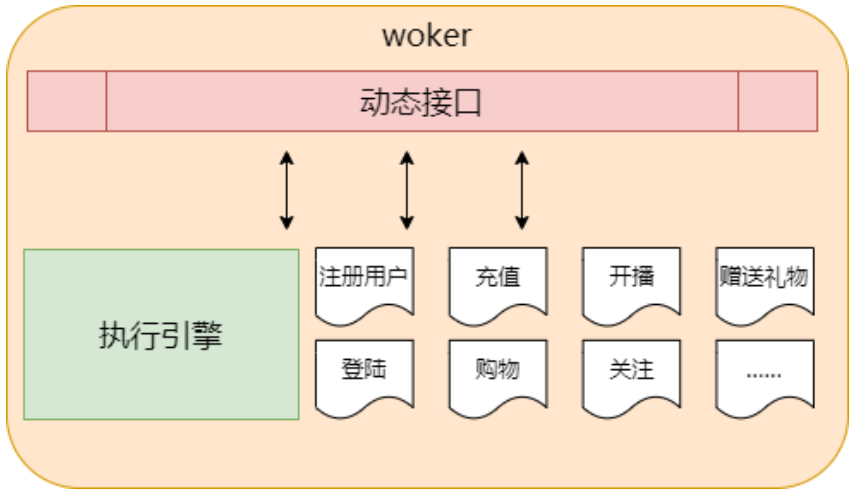


标准流程模型让一份脚本同时实现全链路造数+接口测试+应用拨测+性能测试

/04

云原生之路

线程隔离模式遇到一些问题



- 直播推流
- 压测脚本
- WebSocket推送
- 其他长时间执行的测试任务

用例间隔离

① 脚本代码对系统的影响

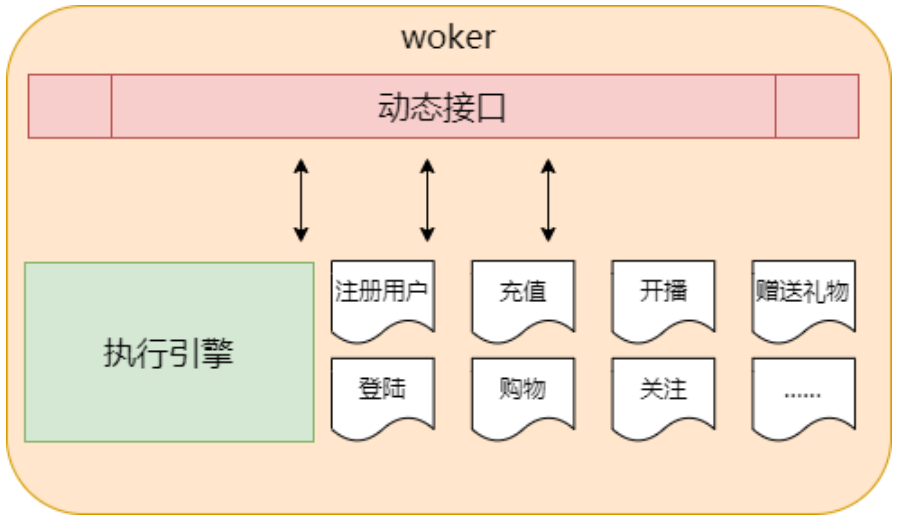
② 对系统资源的占用

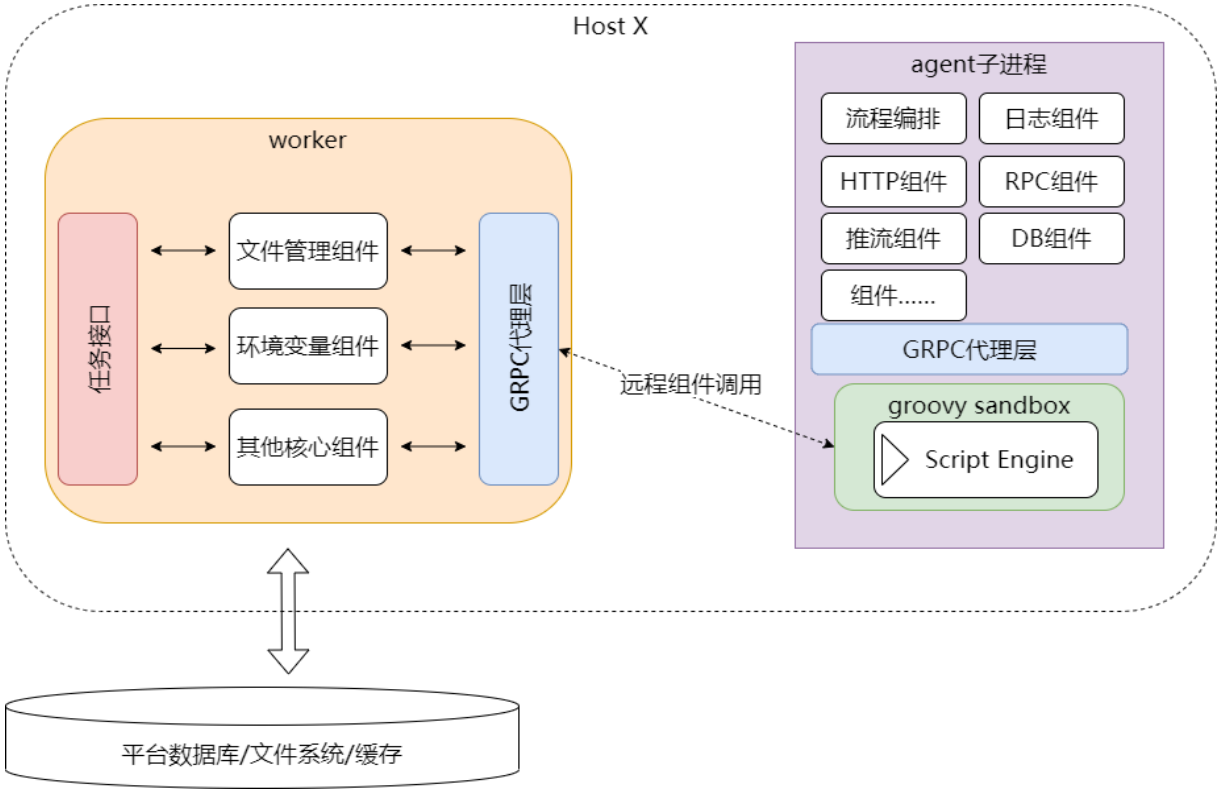
系统对用例的影响

① 主机资源，影响任务稳定行

② 发布导致长时间执行任务中断

从线程隔离到进程隔离的重构





也发掘出了一些令人激动的新特性：

动态类库加载

```
java -cp xxxxxx.jar;
```

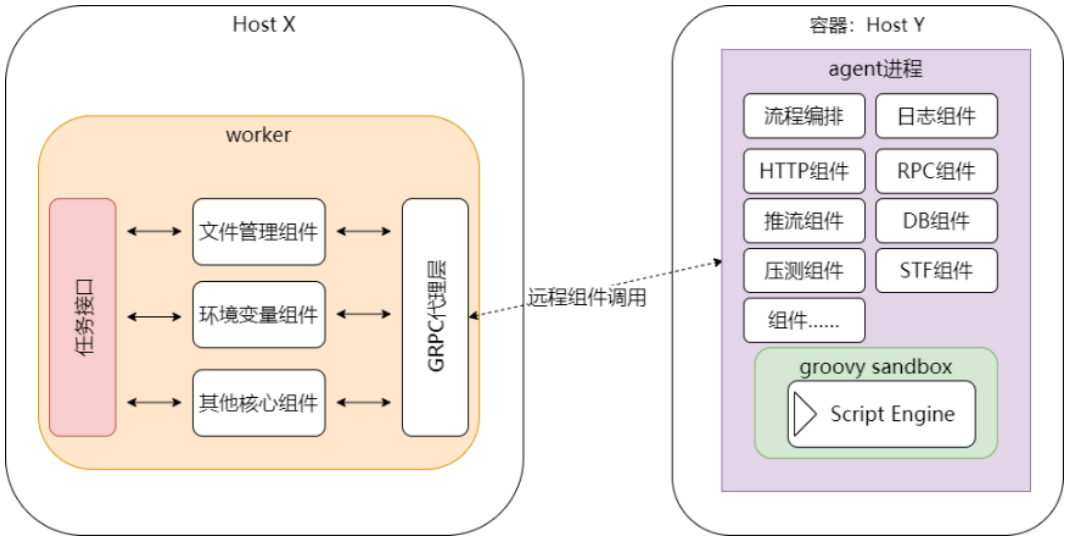
低代码Debug

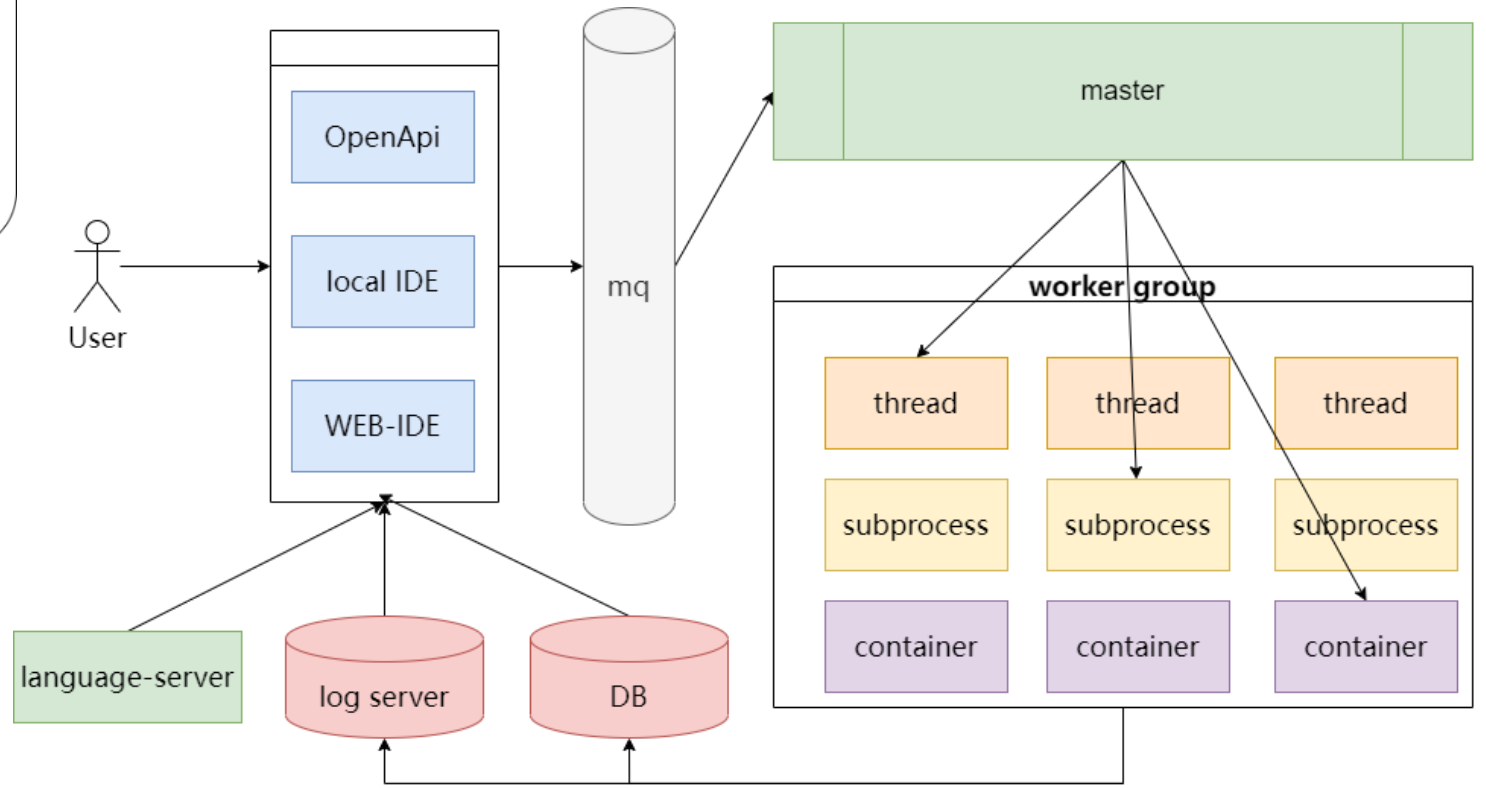
```
jdwp=transport=dt_socket,address=*:8080
```

Serverless

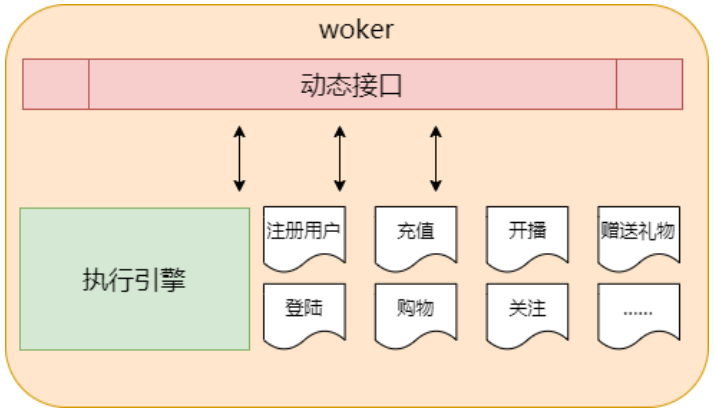
多语言混合编排

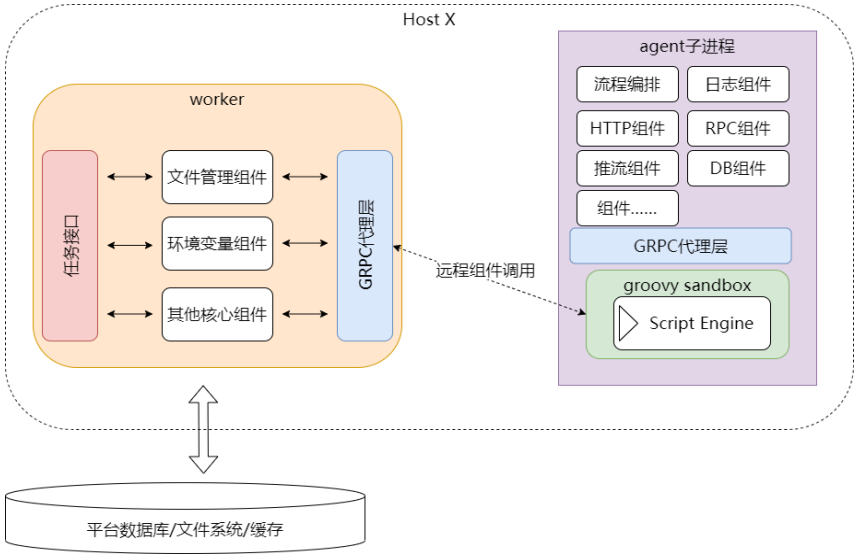
向ServerLess Job演进

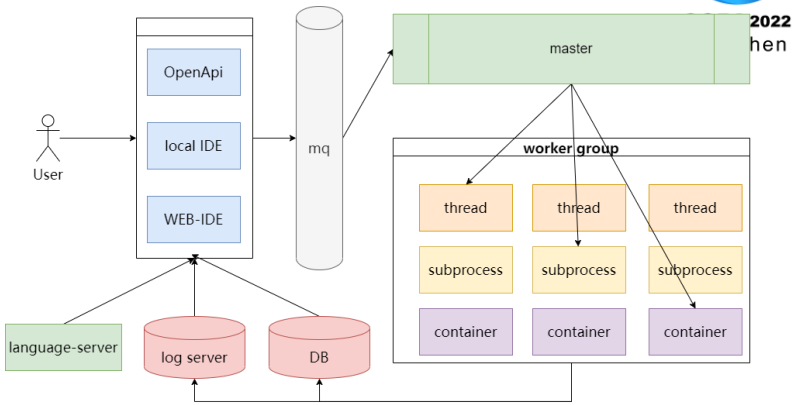


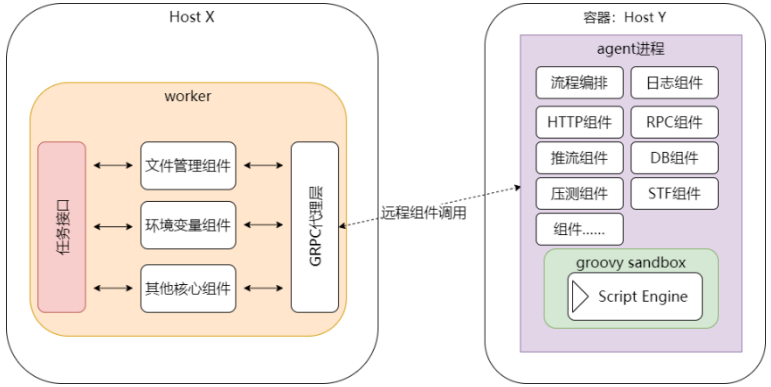


演进之路









线程隔离

进程隔离

镜像/容器隔离

/05

总结与展望

体系概览

可观测

- metric 上报/告警
- log采集
- trace注入
- 任务调度
- 状态管理
- 资源管理

场景接入

- 全链路数据构造
- 接口测试
- 应用拨测
- 性能测试
- 全链路集成测试
- CI/CD集成

开发&使用

- web-ide
- 本地调试
- 语言服务器
- ide插件
- 类库加载
- 运行时profiling

低代码生成

- 环境配置
- 组件配置
- 模板接口抽象
- 动态代码生成
- 动态界面生成
- 动态接口生成

部署&运行

- 源码托管
- 脚本编译
- 异常定位
- 语法分析转换
- 语法级安全控制
- 多语言混合编排

serverless运行

- 负载均衡
- 自动部署
- 线程级隔离&运行
- 节点注册
- 进程级隔离&运行
- 镜像/容器隔离&运行

工具&平台的价值落点?

效率

语言语法层面抽象、低代码生成

标准

定义标准流程模型、组件模型

开放

原生语言语法

宣传

低代码测试引擎技术交流(712523537)

虎牙一直践行“技术驱动内容”，

在**实时内容创作与直播互动技术**上持续创新，

为业务赋能

- 发布直播数字人和小程序开放平
- 4K超分、AI打点实时回放功能上
- 推出虚实同台互动直播

Thanks

开放运维联盟

高效运维社区 **荣誉出品**

DevOps 时代