CPP-Summit 2020

中兴契约测试规模化落地实践

张晓龙
资深架构师

议程

1 **契约测试介绍**

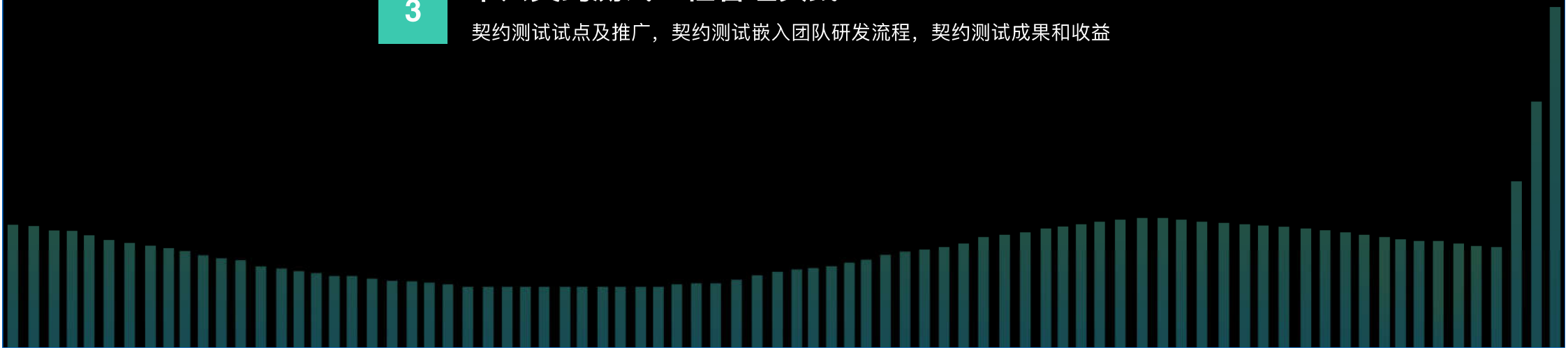
契约测试的背景、概念和开源框架

2 **中兴契约测试技术实践**

契约测试框架 Coral（自研），契约测试用例开发套路，契约测试最佳实践

3 **中兴契约测试工程管理实践**

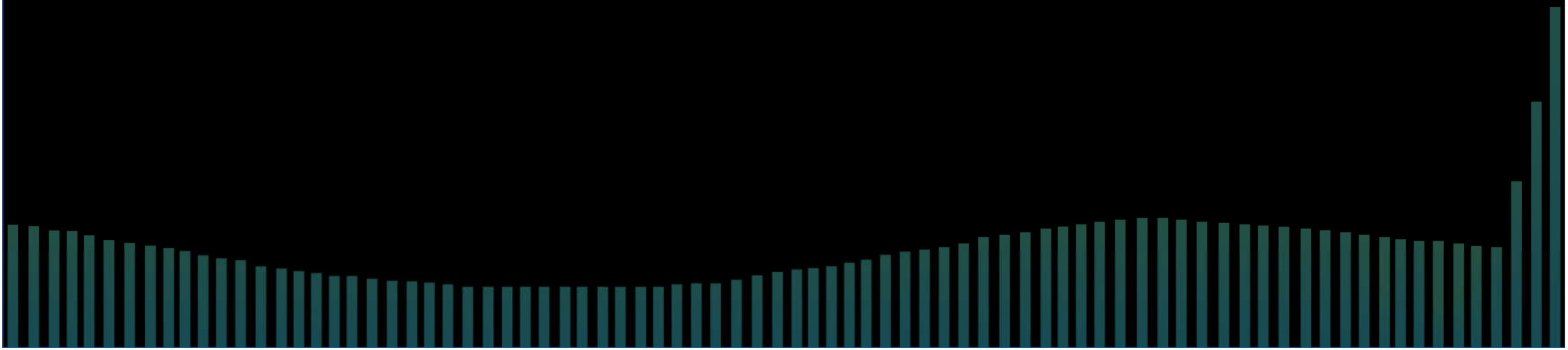
契约测试试点及推广，契约测试嵌入团队研发流程，契约测试成果和收益



01

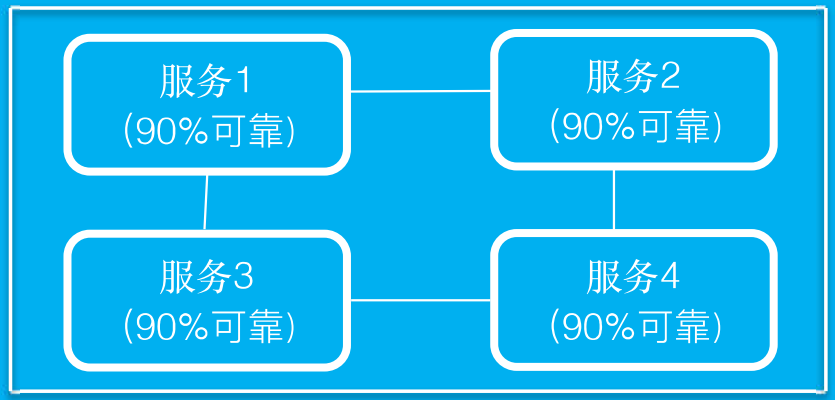
契约测试介绍

契约测试的背景，概念和开源框架

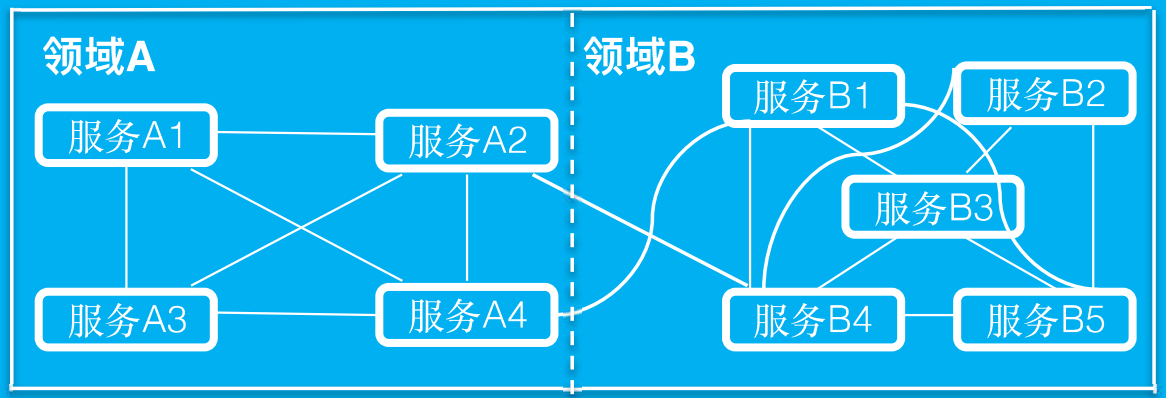


系统可靠性

线性系统

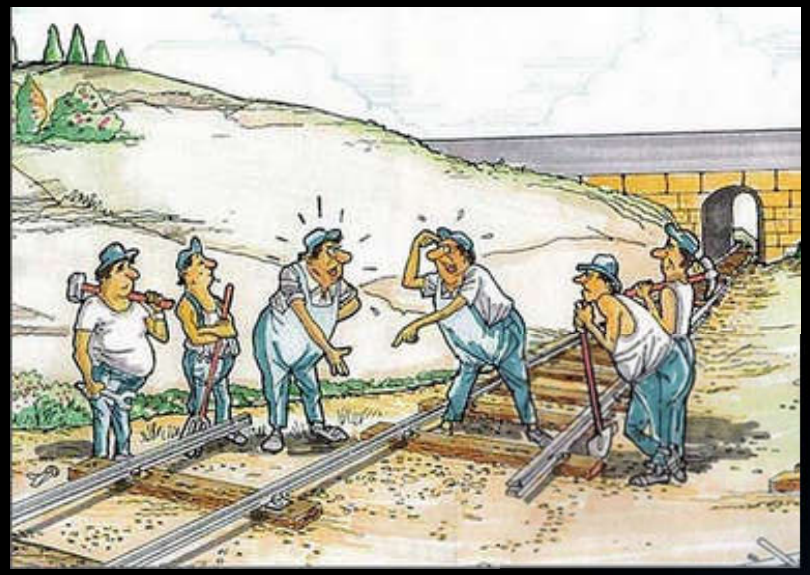


大型微服务系统



- 线性系统（即复杂性随规模呈线性增长）的可靠性等于组成它的各个服务的可靠性之乘积，即系统整体的可靠性（90%×90%×90%×90%=65.61%）低于任一服务的可靠性（90%）
- 一个大型微服务系统，每一个服务的可靠性都对系统整体的可靠性有着非常重要的影响，同时各个服务之间的依赖关系也会对系统的可靠性产生显著影响

如何提高可靠性？



尽早联调

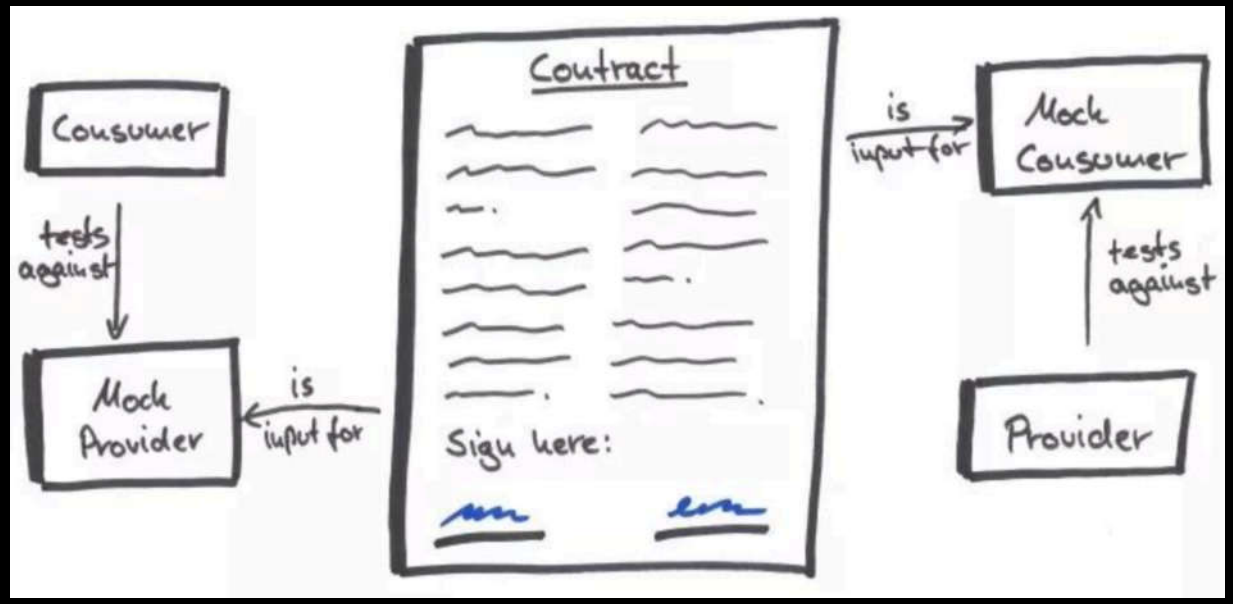
契约测试

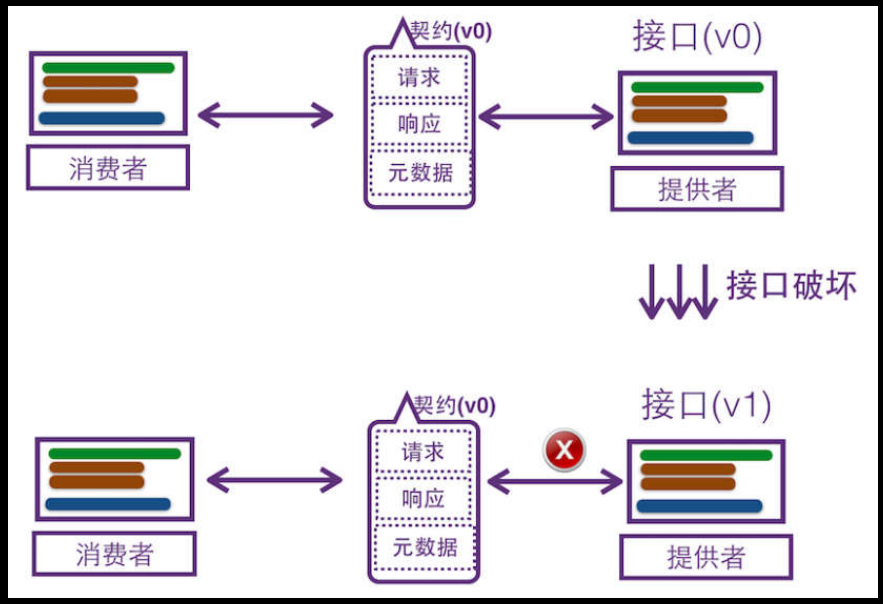
轻量化集成

契约

将契约文字刻写在器皿上（纸上），盖上章（按手印），就是为了使契文中规定的内容得到多方承认和信守

契约测试





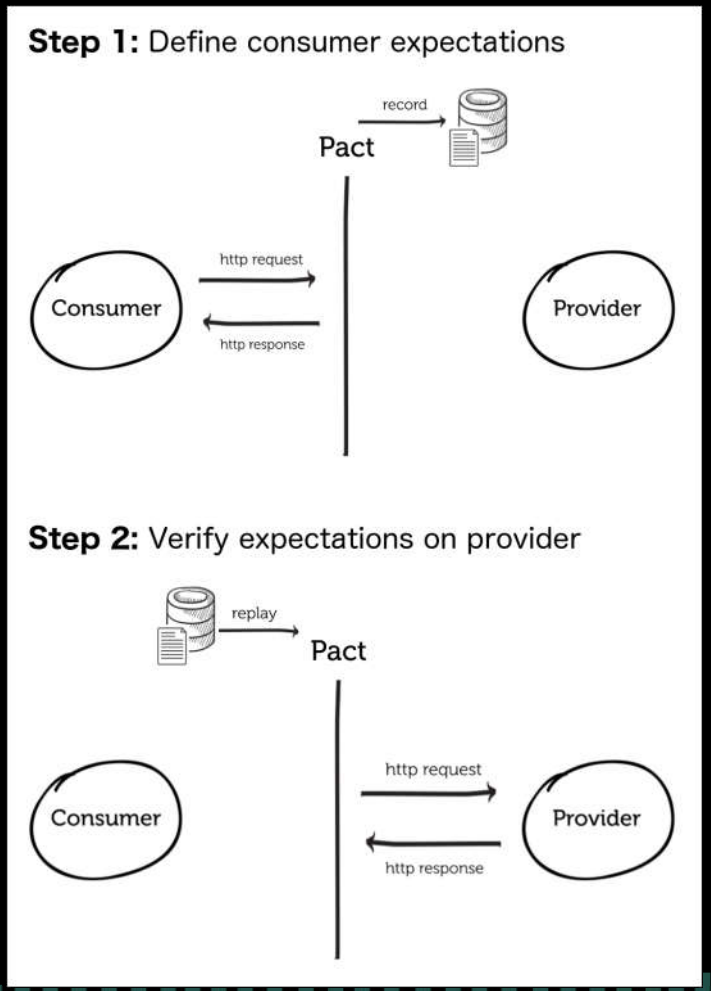
- Consumer 和 Provider 双方可以独立的进行预集成测试，将本来在联调中才能触发的问题前移
- 契约测试守护接口的正确性，一旦接口被破坏，会快速反馈出来

开源契约测试框架

契约测试框架	说明	适用场景
Pact	轻量级、支持多种语言应用、支持与 Maven/Gradle 等集成	使用非 Spring Cloud 框架
Spring Cloud Contract	支持基于 JVM 的应用、支持与 Spring 其它组件集成	使用 Spring Cloud 框架

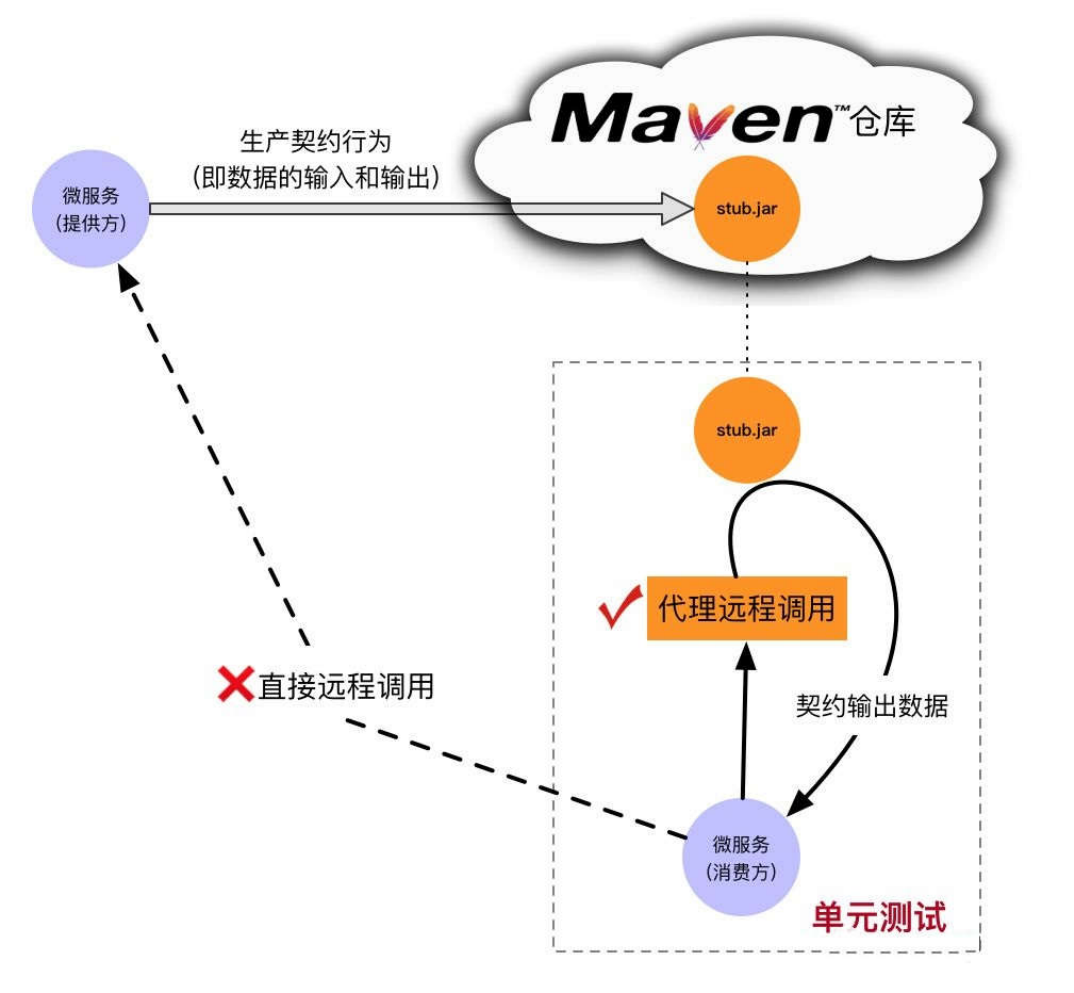
Pact

- 步骤 1： 消费者
  - 编写并运行单元测试（包括对接口的请求参数和预期响应）
  - Pact 代替实际服务提供者（自动）
  - 生成契约文件（自动）
- 步骤 2： 提供者
  - 启动服务提供者
  - Pact 回放契约文件中的请求，验证真实响应是否满足预期（自动）



Spring Cloud Contract

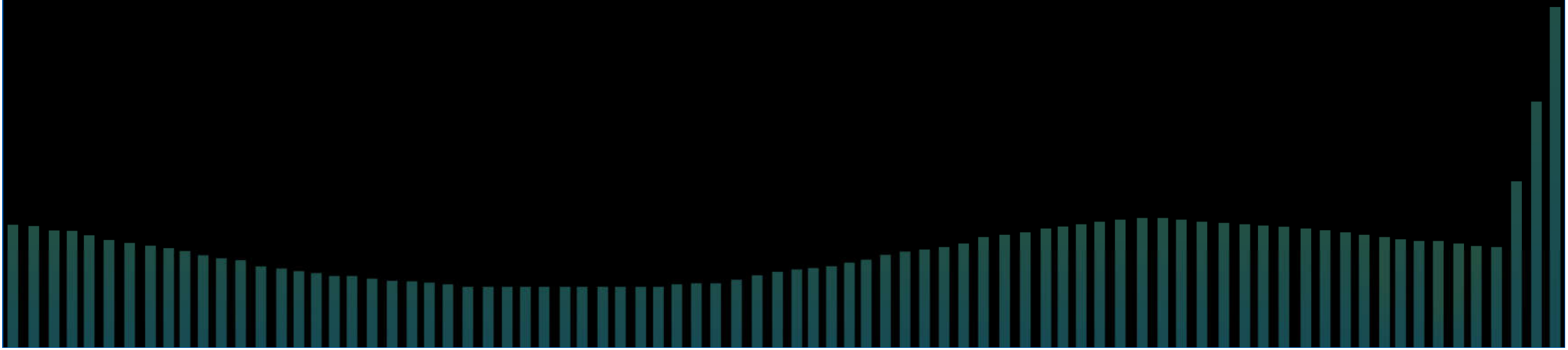
- 步骤 1： 提供者
  - 编写契约
  - 编写测试基类，生成测试用例
  - 运行测试用例，验证真实响应是否满足预期（自动）
  - 发布 stub.jar 包
- 步骤 2： 消费者
  - 编写测试用例
  - 通过注解指定依赖的 stub.jar 包
  - 运行测试用例，验证外部服务正常



02

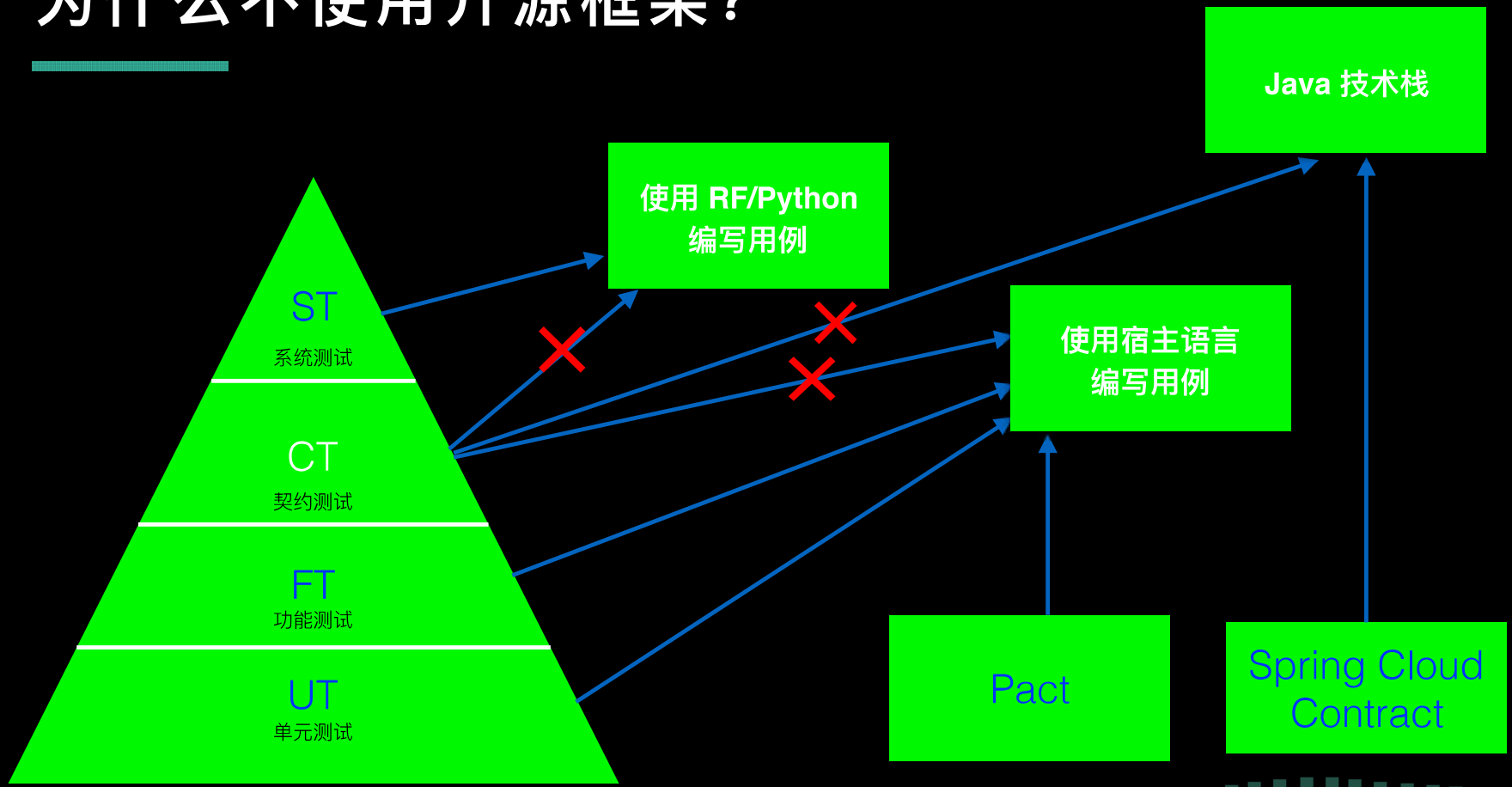
中兴契约测试技术实践

契约测试框架 Coral（自研），契约测试用例开发套路，契约测试最佳实践

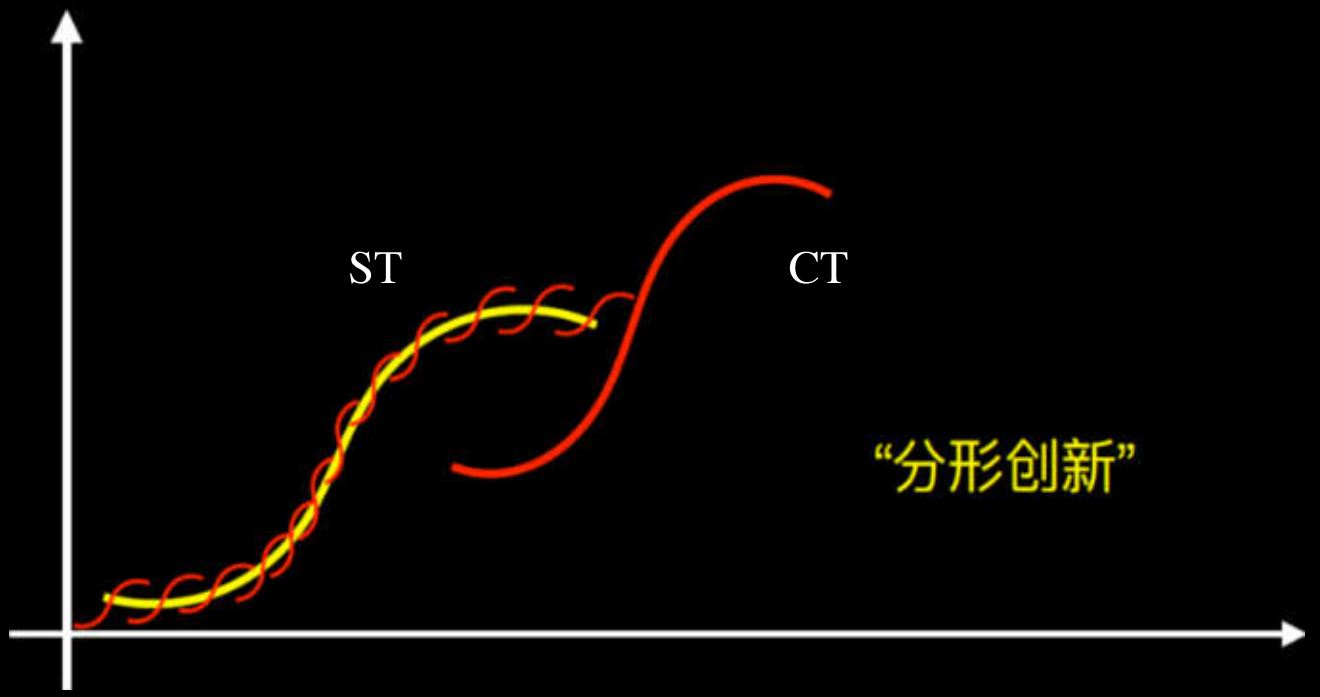


契约测试框架选型

为什么不使用开源框架?

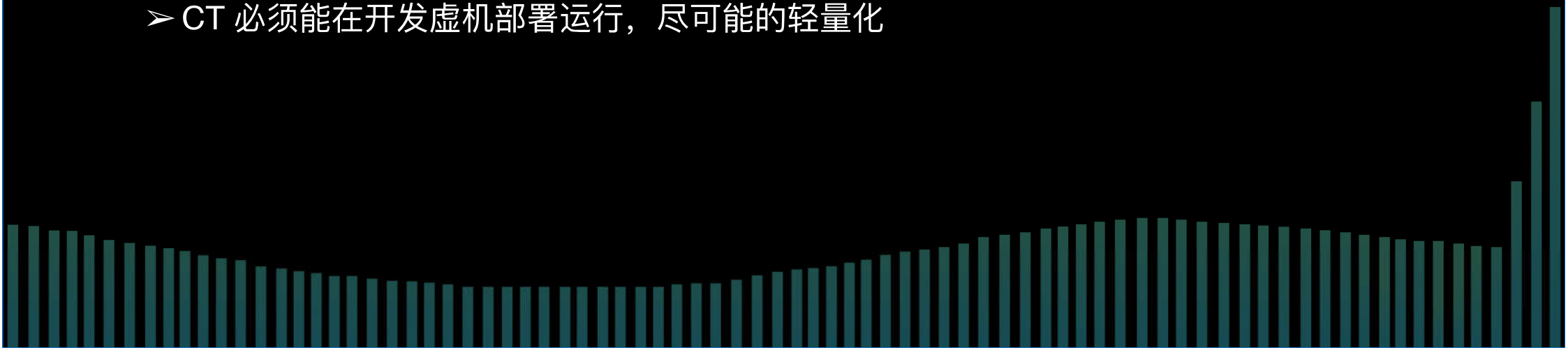

Java 技术栈
使用 RF/Python
编写用例
使用宿主语言
编写用例
ST
系统测试
CT
契约测试
FT
功能测试
UT
单元测试
Pact
Spring Cloud
Contract

契约测试的定位



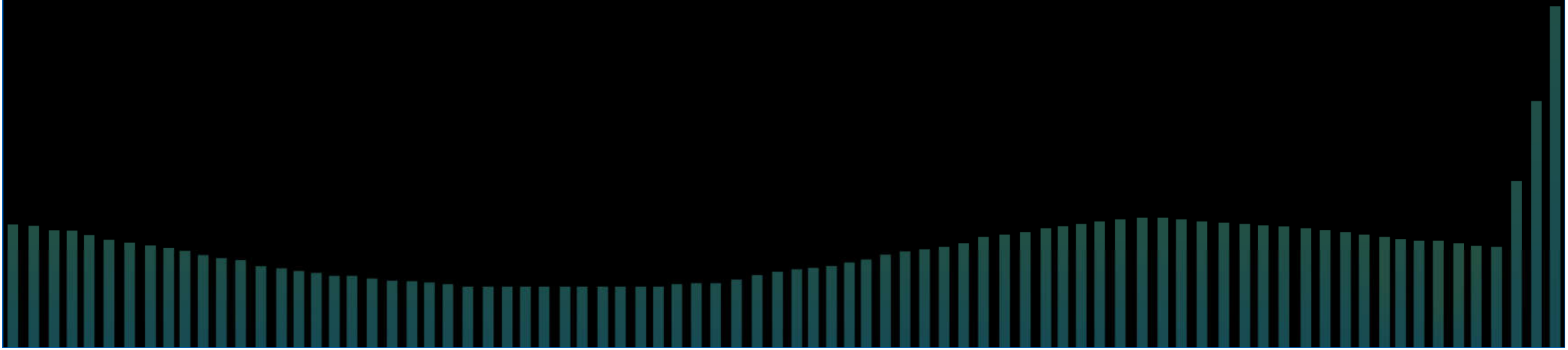
领域诉求

- CT 从 ST 分形而来，SUT 的基础设施尽可能是真实的
- 除过服务契约，还要有流程契约
- 无码化契约测试用例开发，使得懂点业务的同学就可以参与，具有普世价值
- CT 必须能在开发虚机部署运行，尽可能的轻量化



为什么不基于开源框架二次开发?

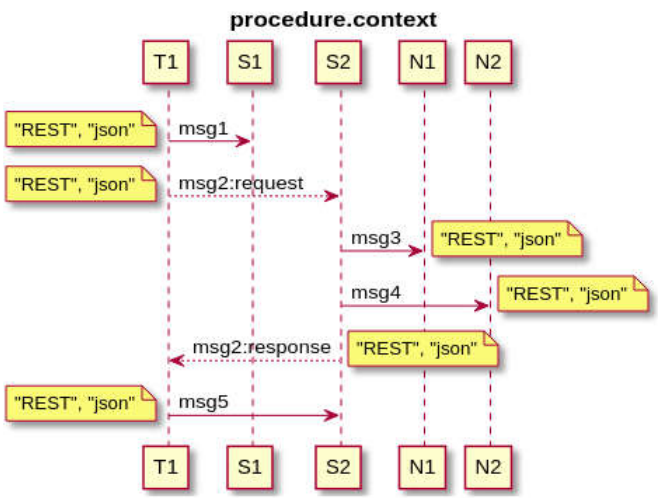
- 技术栈不同
- 能够直接复用的特性不多



Coral 契约测试介绍

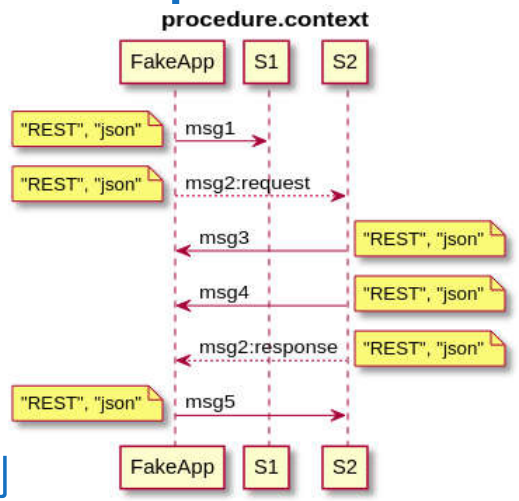
核心设计：流程契约&流程契约实例

```
@startuml

title procedure.context

T1 -> S1: msg1
note left: "REST", "json"

T1 --> S2: msg2:request
note left: "REST", "json"

S2 -> N1: msg3
note right: "REST", "json"

S2 -> N2: msg4
note right: "REST", "json"

S2 --> T1: msg2:response
note right: "REST", "json"

T1 -> S2: msg5
note left: "REST", "json"

@enduml
```



流程契约

```
@startuml

title procedure.context

FakeApp -> S1: msg1
note left: "REST", "json"

FakeApp --> S2: msg2:request
note left: "REST", "json"

S2 -> FakeApp: msg3
note right: "REST", "json"

S2 -> FakeApp: msg4
note right: "REST", "json"

S2 --> FakeApp: msg2:response
note right: "REST", "json"

FakeApp -> S2: msg5
note left: "REST", "json"

@enduml
```



流程契约实例

核心设计：服务契约&服务契约实例

```
swagger: '2.0'
info:
  title: create_rcslist_succ.template
  description: The service contract of the create_rcslist_succ msg
  version: 1.0.0
paths:
  /api/v1/namespaces/{namespace}/rcslist:
    post:
      summary: create rc list of the namespace
      description: create some rc objs in the namespace       parameters:
        - name: namespace
          in: path
          description: name of the namespace
          required: true
          type: string
        - name: body
          in: body
          description: rc list
          required: true
          schema:
            $ref: '#/definitions/rclist'
      responses:
        '200':
          description: operate successfully
          schema:
            type: object
            properties:
              status:
                type: string
```

服务契约

实例化

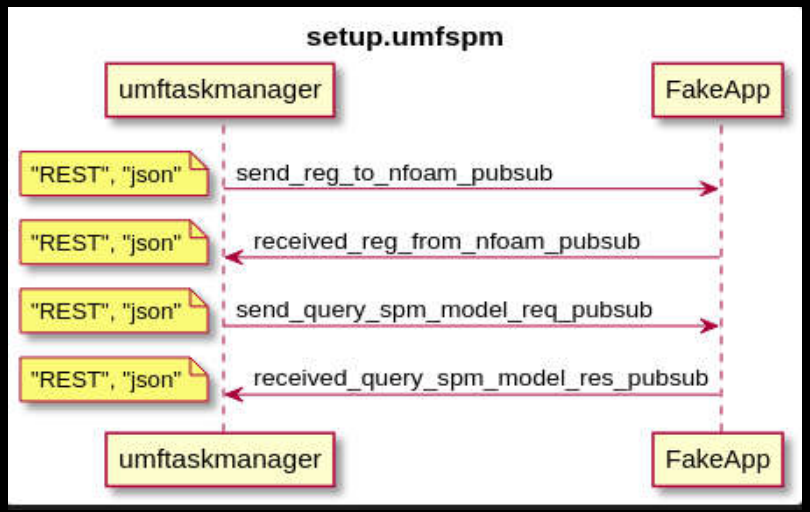
```
{
    "base_info":{
        "protocol":"http",
        "port":"PROVIDER_PORT"
    },
    "request":{
        "method":"GET",
        "url":"/foobar",
        "headers":{
            "Content-Type":"application/json"
        },
        "body":{
        }
    },
    "response":{
      "status": 200,
        "headers": {
            "Content-Type": "text/plain"
        },
        "body": {
          "status": "${STATUS}"
        }
    },
    "parameters": [
        {
          "name":"STATUS",
          "value":"[foo,foo1,foo2]",
          "default": "foo1"
        }
    ]
}
```

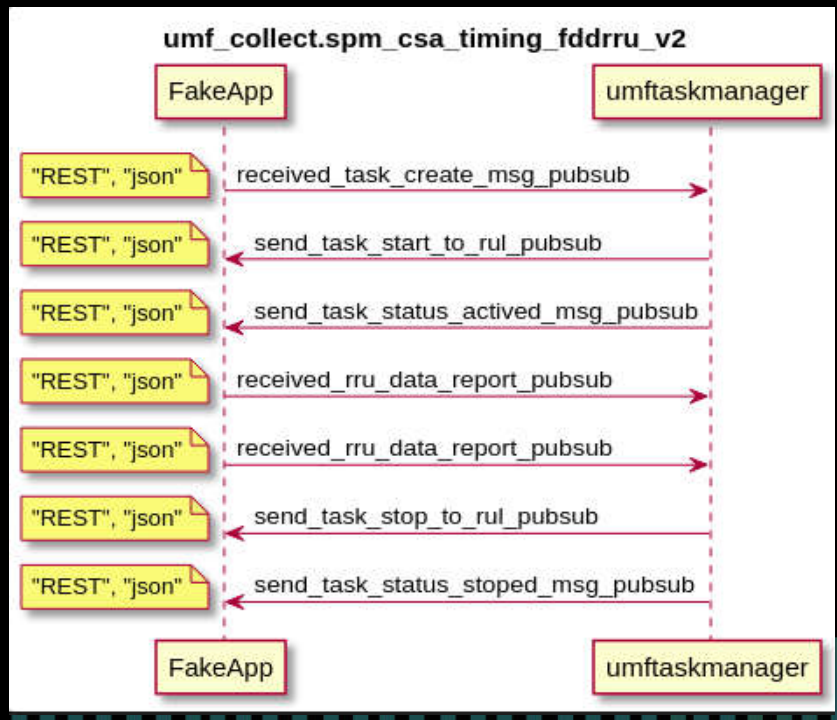
服务契约实例

核心设计：测试套件

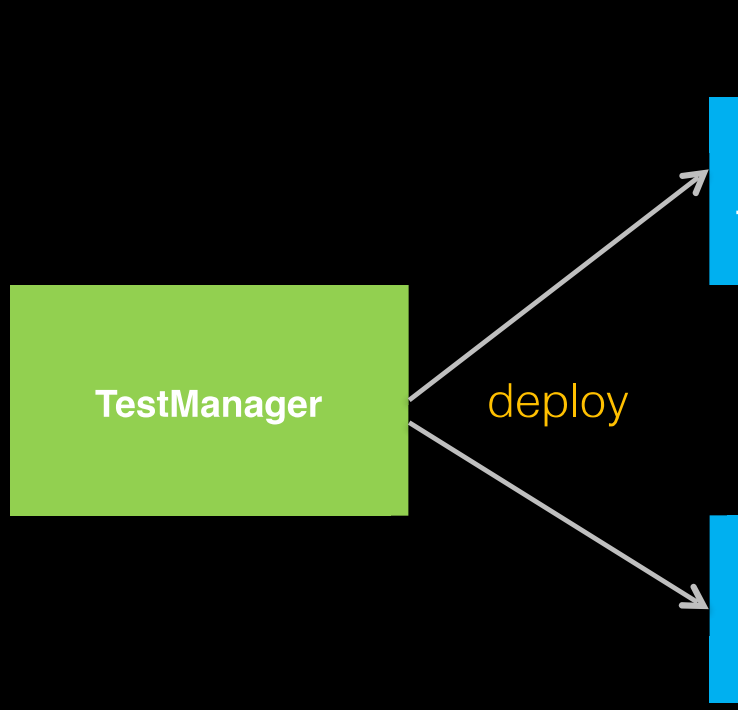
➢ 根据四元组 (procedure, context, domain, tag_exp) 过滤用例集

➢ setup用例





核心设计：SUT 部署



以进程方式部署SUT

```
72  #start monitor
73  exec -c $paas_dir/op-conf/monitor/monitor
74  echo "monitor started"
```

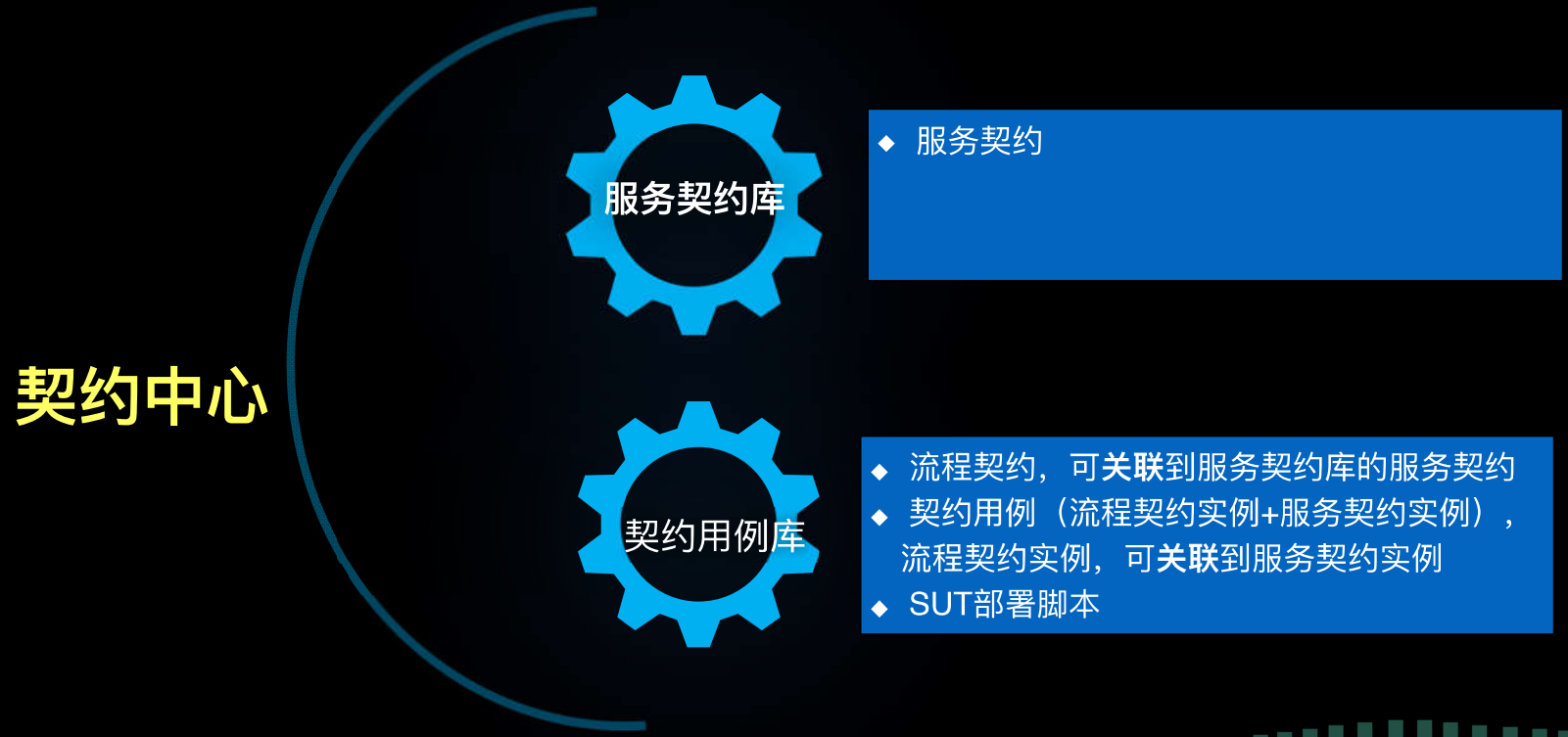
以容器方式部署SUT

```
76  echo "UMF_IMAGE_TAG:$UMF_IMAGE_TAG"
77  echo "UMF_IMAGE_ID:$UMF_IMAGE_ID"
78  docker run --name ${umf_docker_name}
```

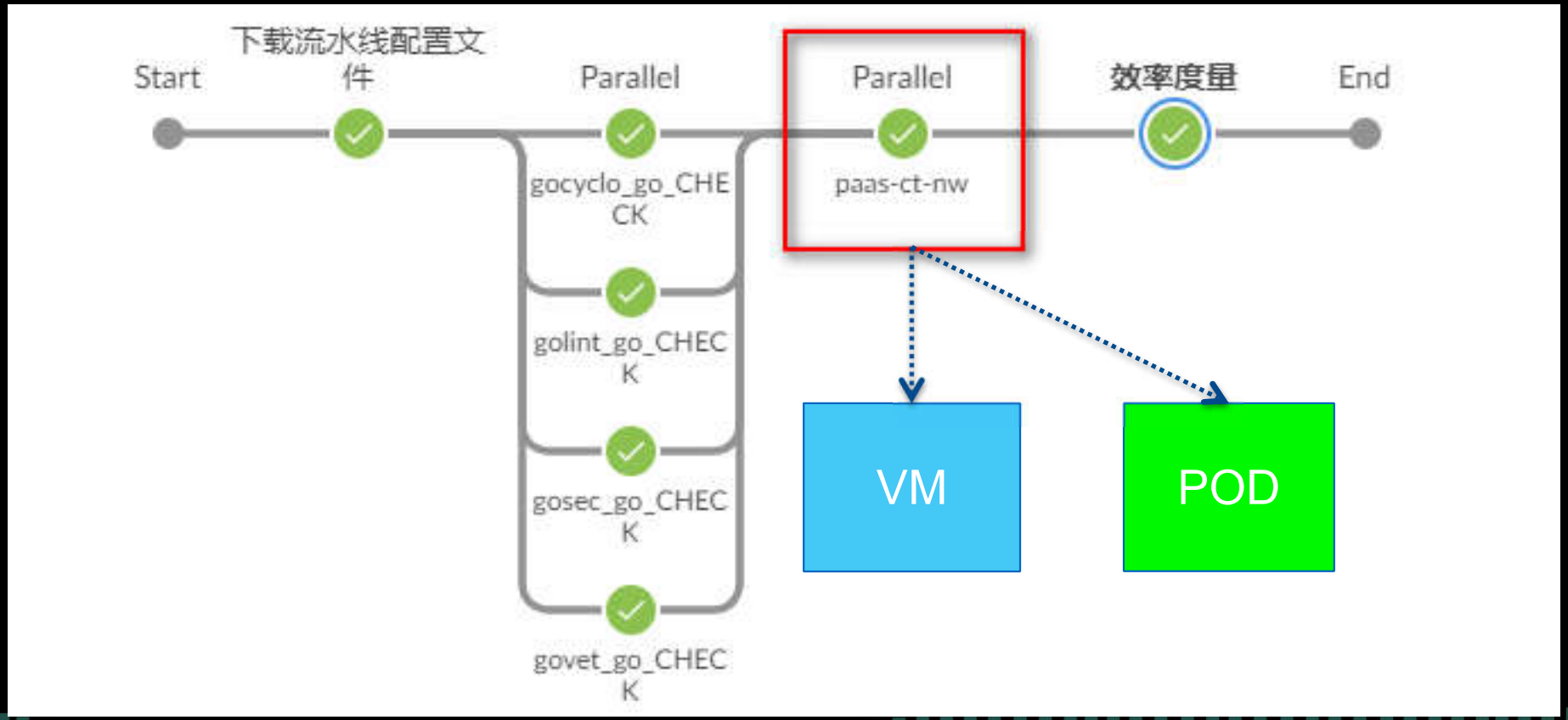
SUT部署脚本调用数据库的部署脚本

```
69  #start pg
70  cd $WORK_DIR/../script/pg
71  ./pg_teardown.sh
72  ./pg_setup.sh $PG_PORT knitter 1234567a knitter
```

核心设计：契约中心


契约中心
服务契约库
契约用例库
◆ 服务契约
◆ 流程契约，可关联到服务契约库的服务契约
◆ 契约用例（流程契约实例+服务契约实例），流程契约实例，可关联到服务契约实例
◆ SUT部署脚本

核心设计：CI 上线


Start
下载流水线配置文件
Parallel
Parallel
效率度量
End
gocyclo_go_CHECK
golint_go_CHECK
gosec_go_CHECK
govet_go_CHECK
paas-ct-nw
VM
POD

核心设计：结构化日志

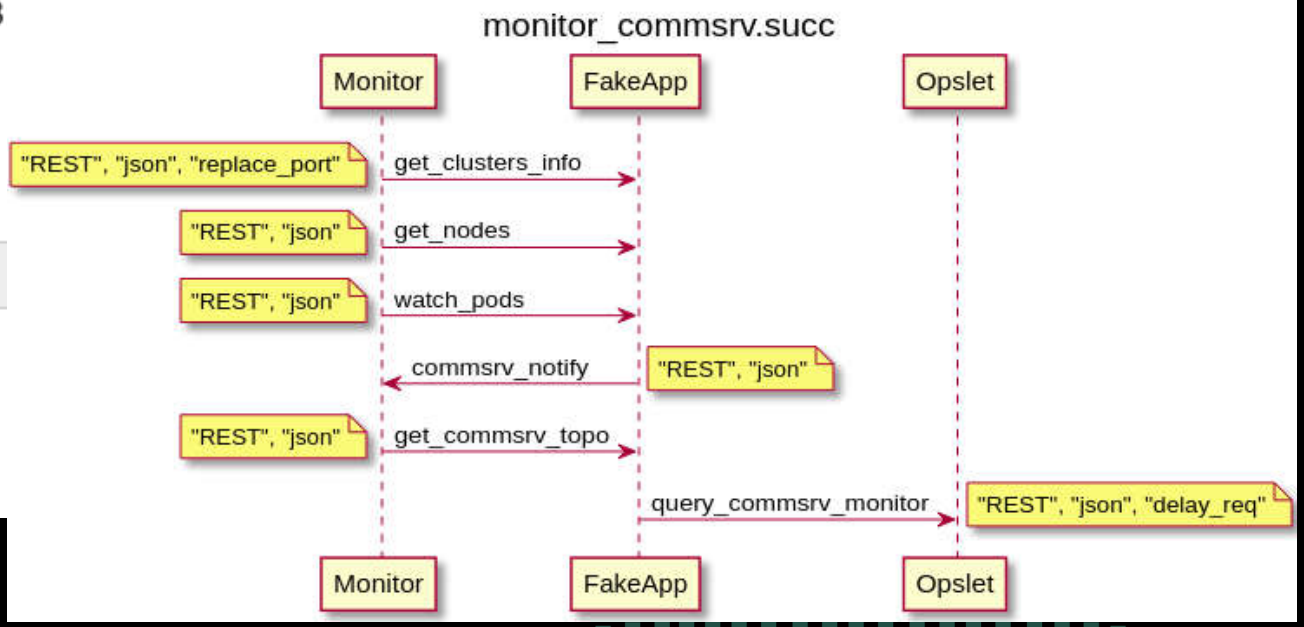
TEST /home/coraldata/testcase/ops/monitor_commsrv/succ/inst/monitor.puml

Full Name: suites.monitor./home/coraldata/testcase/ops/monitor_commsrv/succ/inst/monitor.puml

Start / End / Elapsed: 20201113 14:08

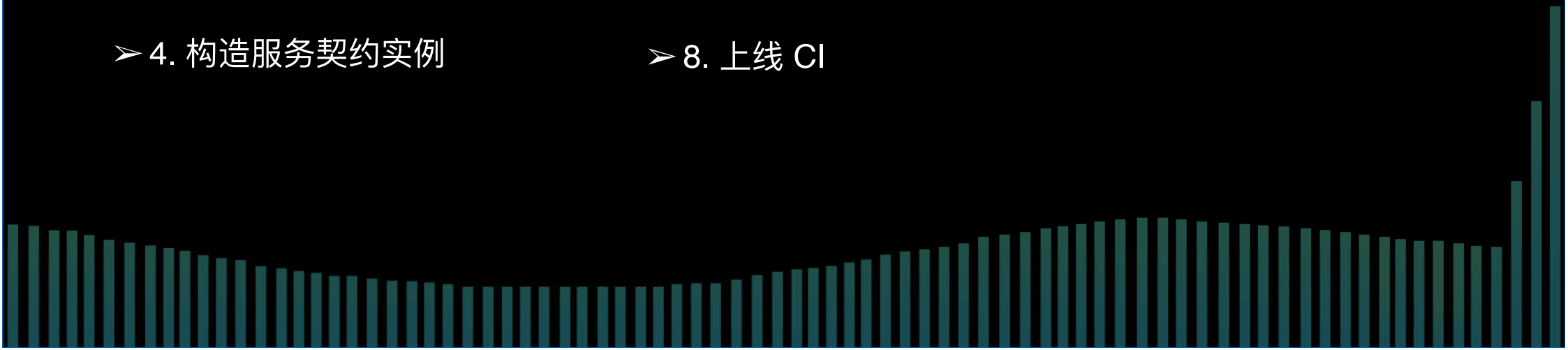
Status: PASS (critical)

- KEYWORD get_clusters_info
- KEYWORD get_nodes
- KEYWORD watch_pods
- KEYWORD commsrv_notify
- KEYWORD get_commsrv_topo
- KEYWORD query_commsrv_monitor



契约测试用例开发步骤

- 1. 编写流程契约
- 2. 编写服务契约
- 3. 生成流程契约实例
- 4. 构造服务契约实例
- 5. 编写 SUT 部署脚本（Once）
- 6. 调试
- 7. 上传契约中心
- 8. 上线 CI



Coral 契约测试框架介绍

契约测试框架 Coral

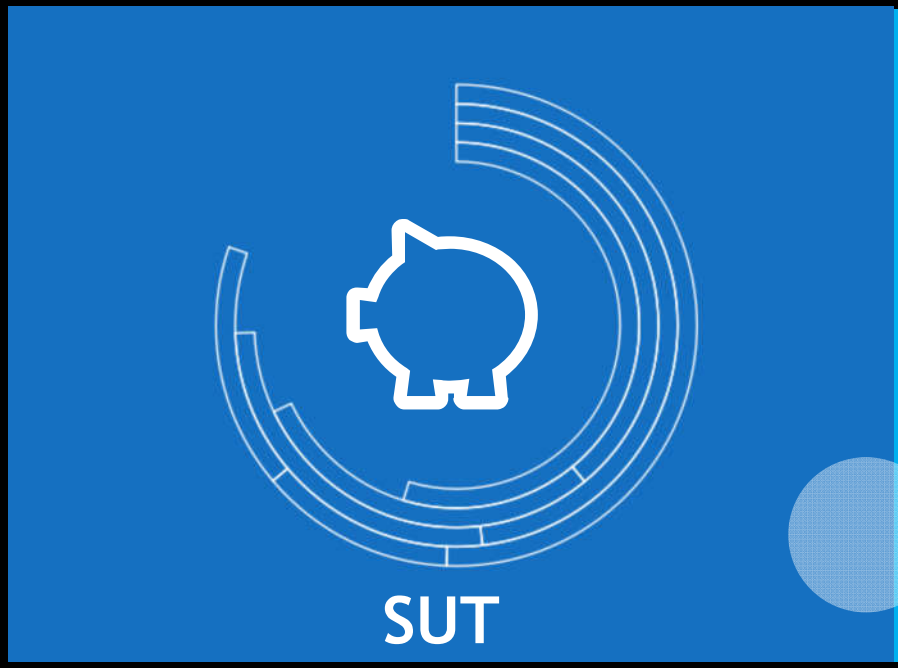
两个角色

三方关系

四项特征

八大价值

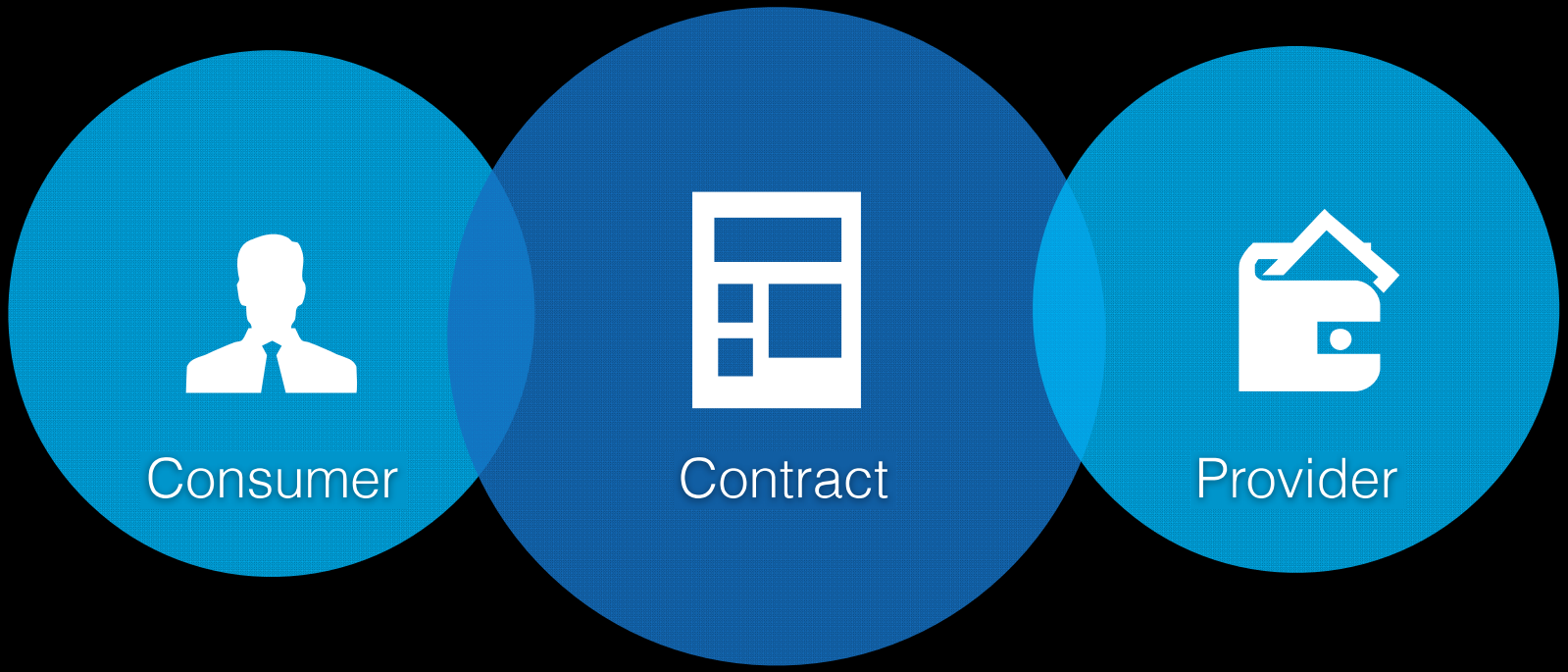
两个角色



被测系统（System Under Test）：被测系统是当前被测试的系统，根据测试系统的不同，SUT 所指代的内容也不同，可以是一个类也可以是整个系统

桩应用：测试中通过 Mock 实现的服务，除过 SUT 之外的所有服务

三方关系



- Consumer 为服务消费者，Provider 为服务提供者，Constract 为两者之间交互的契约
- Consumer 不再依赖 Provider，而是两者都依赖于 Contract

四项特征

轻量级

服务拆分
按需组合

通用性

为 ICT 而生
微服务架构

易用性

一键部署
一键测试

标准化

紧靠业界标准
没有特殊处理

八大价值

低成本回放系统测试
不用搭建系统测试环境
部署便捷
测试运行时间短

增加防火墙厚度
异常容易制造
用例覆盖面广

联调双方解耦
联调双方都依赖于契约
契约测试用例提前上 CI
降低联调成本

测试分层更合理
处于 FT 和 ST 之间专注领域内
ST 变薄专注领域间
用例维护成本降低
TS 根据三原则给用例打标签：场景覆盖到、成本低、最小知识

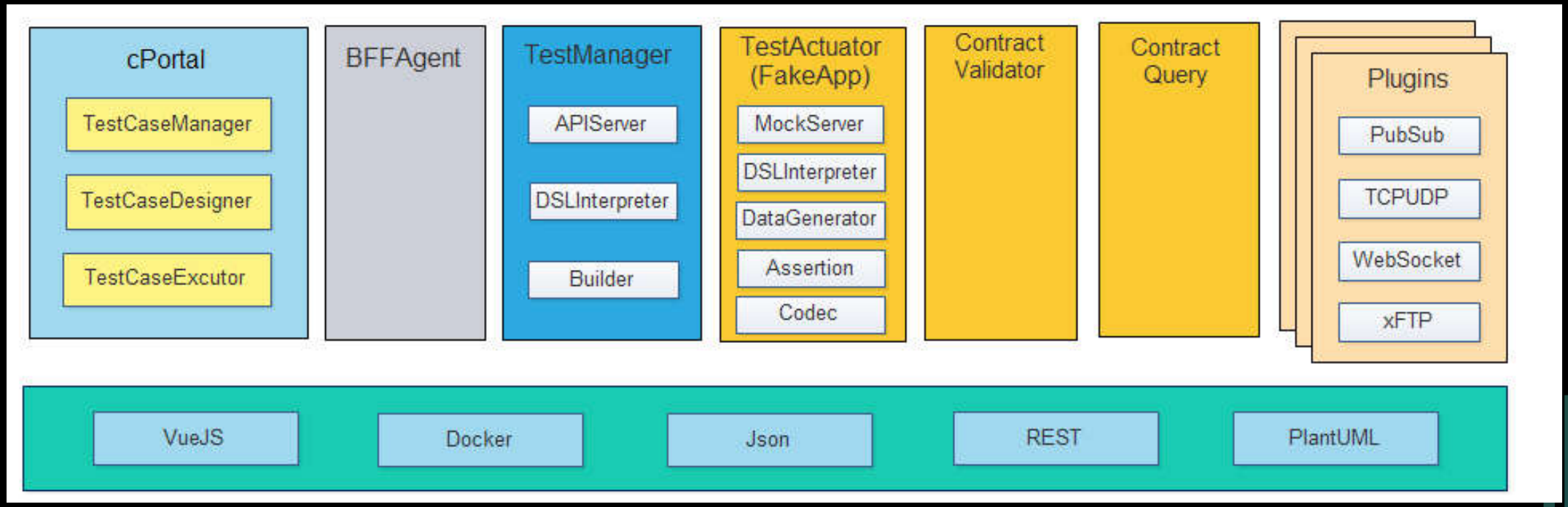
回放产品用户契约
录制产品用户契约
版本发布前在真实环境部署
自动回放用户契约以确保版本质量

用例 CI 与文档 CI
用例 CI 守护用例质量
文档 CI 自动生成 API 文档
版本、用例和文档一致

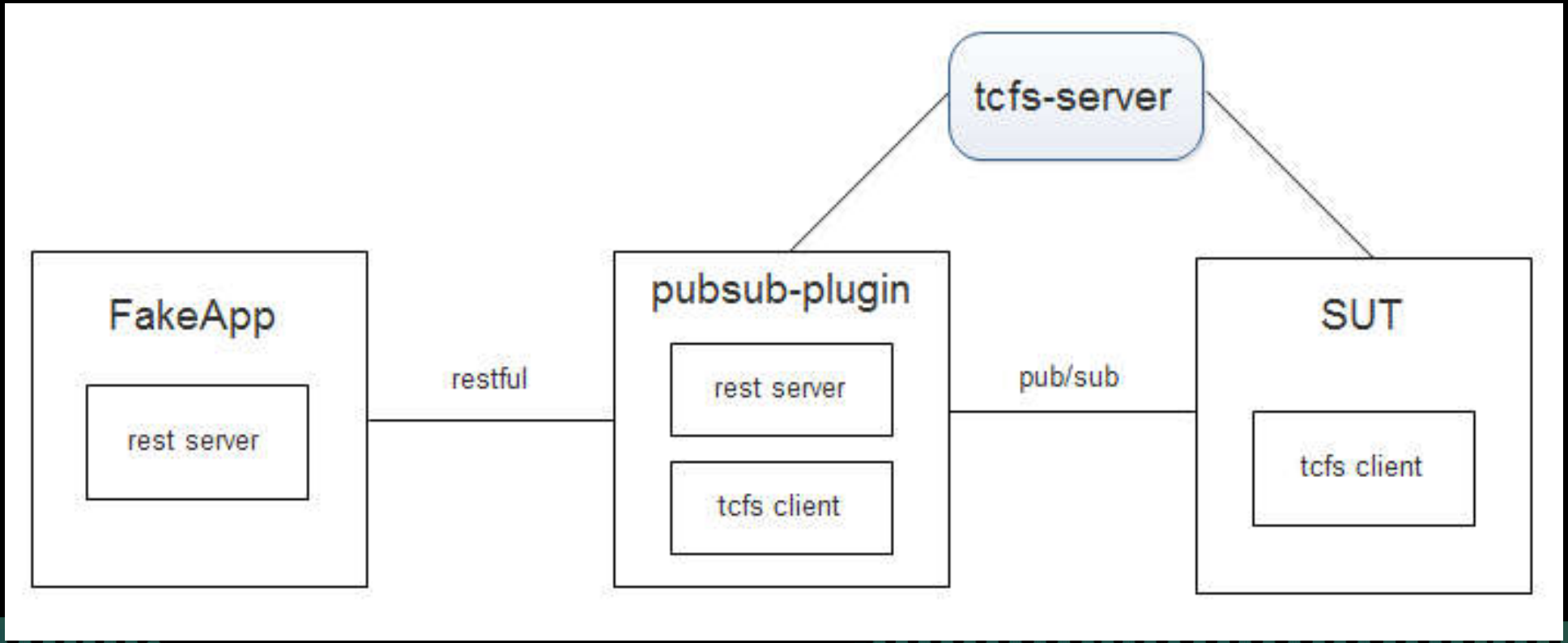
活文档
显式化领域边界行为
文档与代码同步变更
学习领域知识最生动的教材

增加接口变动的安全性和准确性
接口设计约束
接口变动监控
降低验证成本

Coral 微服务架构


cPortal
TestCaseManager
TestCaseDesigner
TestCaseExcutor
BFFAgent
TestManager
APIServer
DSLInterpreter
Builder
TestActuator (FakeApp)
MockServer
DSLInterpreter
DataGenerator
Assertion
Codec
Contract Validator
Contract Query
Plugins
PubSub
TCPUDP
WebSocket
xFTP
VueJS
Docker
Json
REST
PlantUML

插件设计

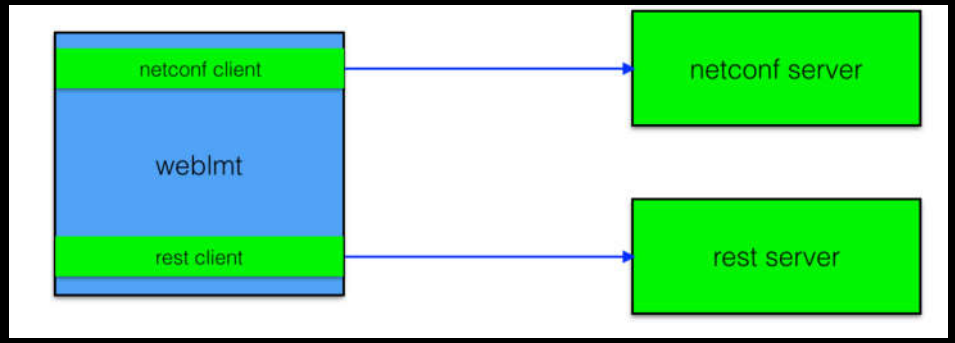

tcfs-server
FakeApp
rest server
restful
pubsub-plugin
rest server
tcfs client
pub/sub
SUT
tcfs client

Coral 契约测试最佳实践

最佳实践一：NETCONF 协议

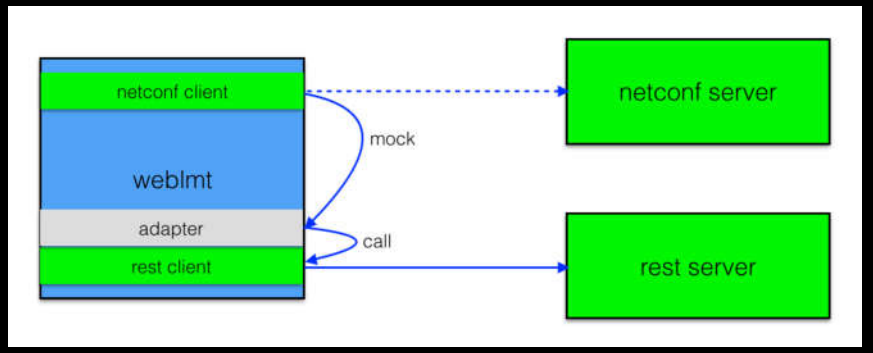
挑战

➢ SUT 依赖的协议重，模拟成本太高，真实部署太耗资源



解决方案

➢ 当 CT 模式时偷梁换柱（SUT Mock）



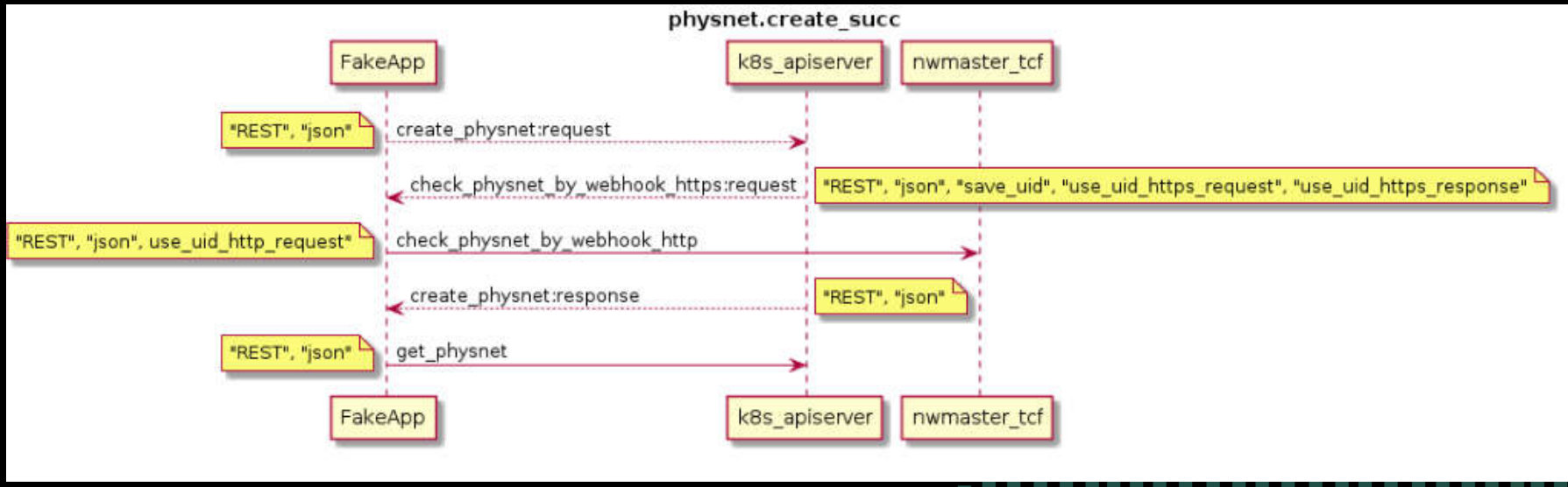
最佳实践二：K8S

挑战

➢ K8S 行为比较复杂，模拟成本太高

解决方案

➢ 将 APIServer 和 ETCD 作为 SUT 的一部分部署起来



最佳实践三：内存数据库

挑战

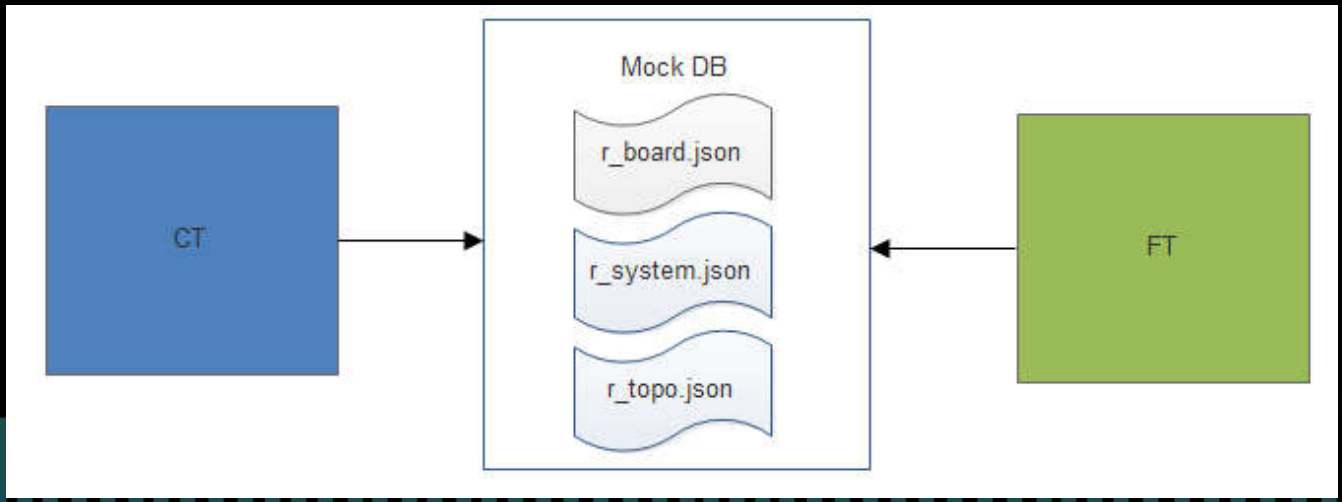
➢ 内存数据库表之间关系复杂，普通开发人员很难维护

➢ 内存数据库升级成本大

解决方案

➢ 数据库对象通过简单工厂模式来构造，当 CT 模式时选择不同的变化方向

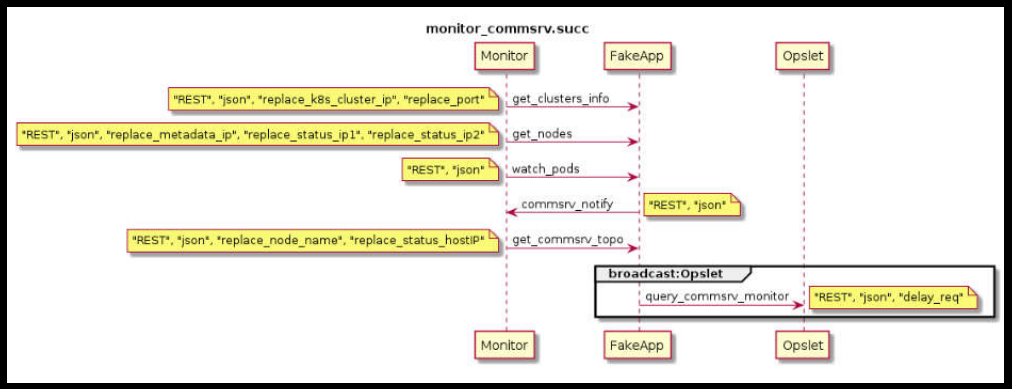
➢ 每张表都是独立的，用 json 来表达，普通开发人员易维护



最佳实践四：SUT 分布式

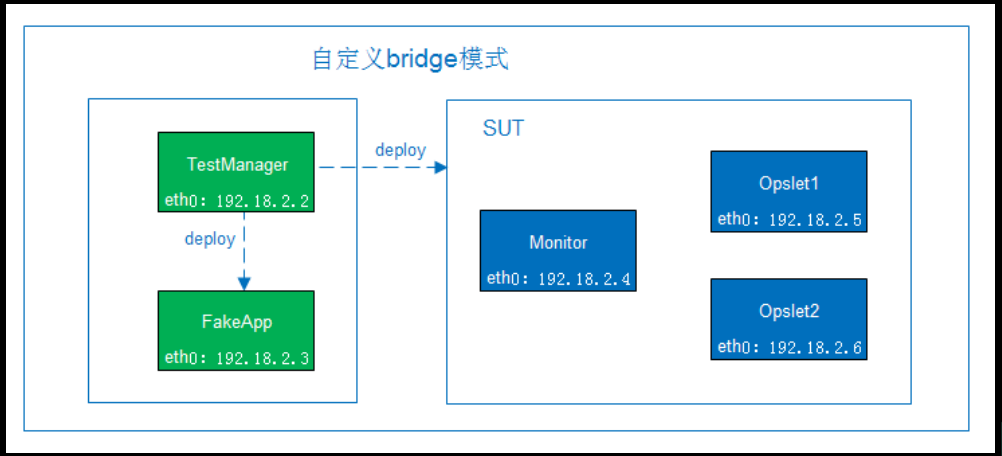
挑战

➢ SUT 中部分服务单实例，部分服务多实例



解决方案

➢ 支持自定义 bridge 模式，为所有容器分配 IP



最佳实践五：并发

挑战

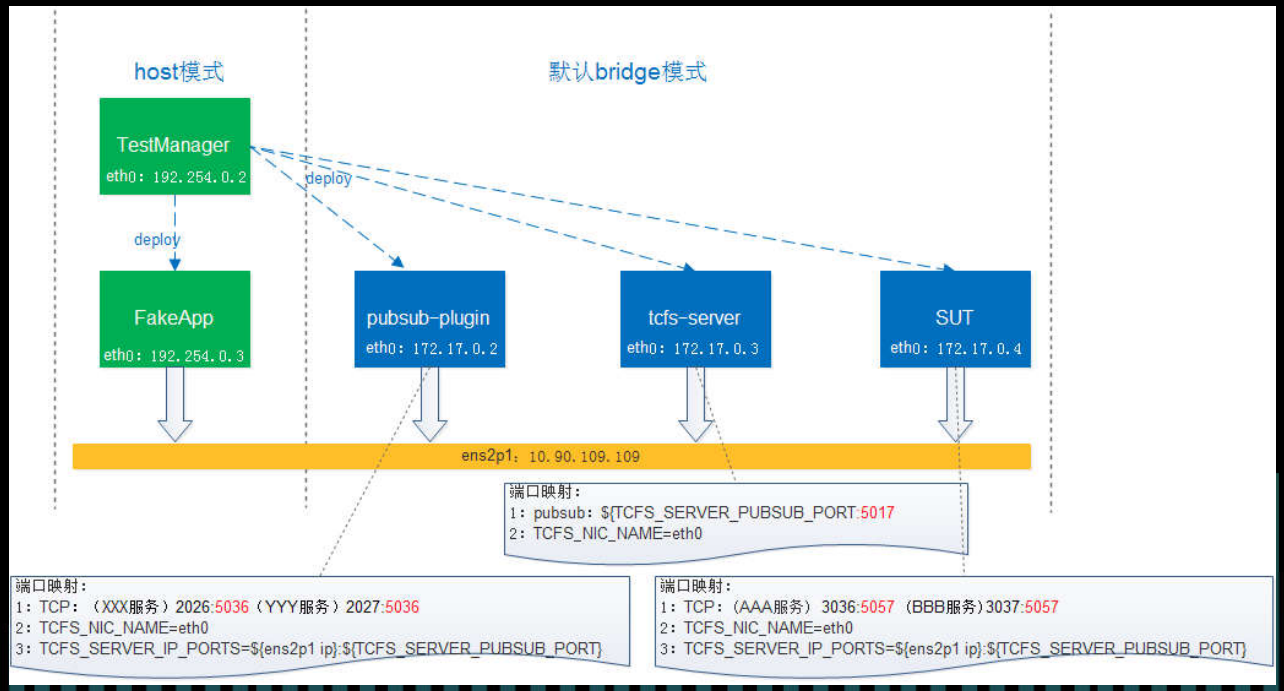
➢ 在一个单板内并发的跑多组 UT，多组 FT，多组 CT，如何隔离？

➢ 电信微服务平台中某些端口是写死的，比如协议栈的 TCP 端口和消息总线的 Serer 端口，如何并发多组 CT？

解决方案

➢ CT 容器通过命名空间，精准管理其生命周期，不能误杀 UT 容器和 FT 容器，也不能误杀其他组的 CT 容器

➢ 通过容器默认的 bridge 模式来隔离端口，Coral 不感知



最佳实践六：二进制码流

挑战

➢ 电信微服务总线发送消息的 payload 为 json 或二进制码流，二进制码流如何表达和编码?

```
{
    "Msgbrief": {
        "SessionType": "RUL_REROUTE_SESSION",
        "MsgId": 1,
        "SessionInst": "00000005",
        "Version": 64,
        "Length": 100,
        "Priority": 2
    },
    "Payload": "[89 01 00 01 00 00 00 00 00 01 3d 10 2c ff]",
    "parameters": []
}
```

解决方案

➢ 自定义标准：通过以 “[” 开始以 “]” 结束的十六进制码流字符串来表达二进制码流

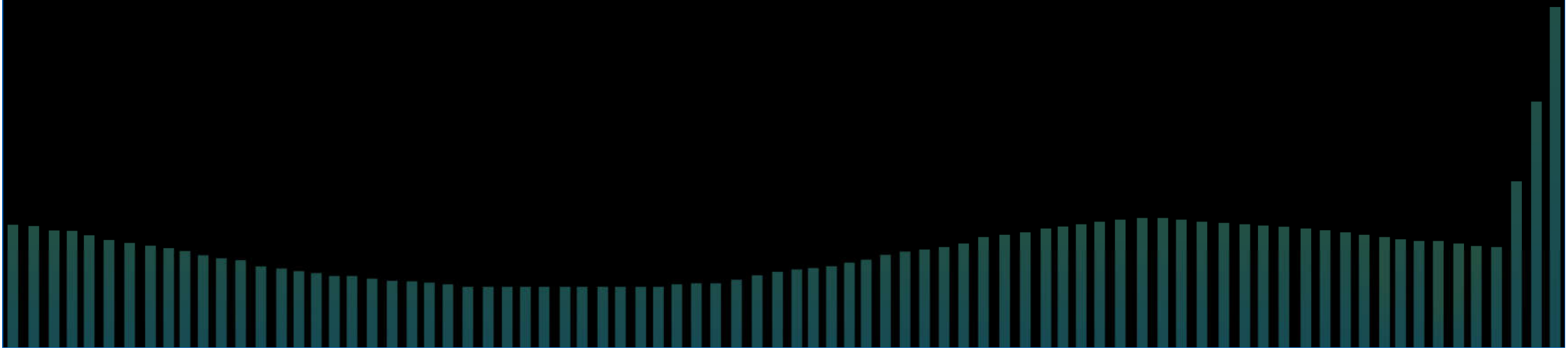
➢ FakeApp 发送二进制码流： a. 去掉 “[” 和 “]” ； b. 去掉字符串双引号； c. base64 编码； d. 序列化；

➢ FakeApp 接收二进制码流：a.反序列化； b. base64 解码； c. 转成十六进制码流字符串（actual）； d. 取出该服务契约实例中的十六进制码流字符串（expect）进行断言

03

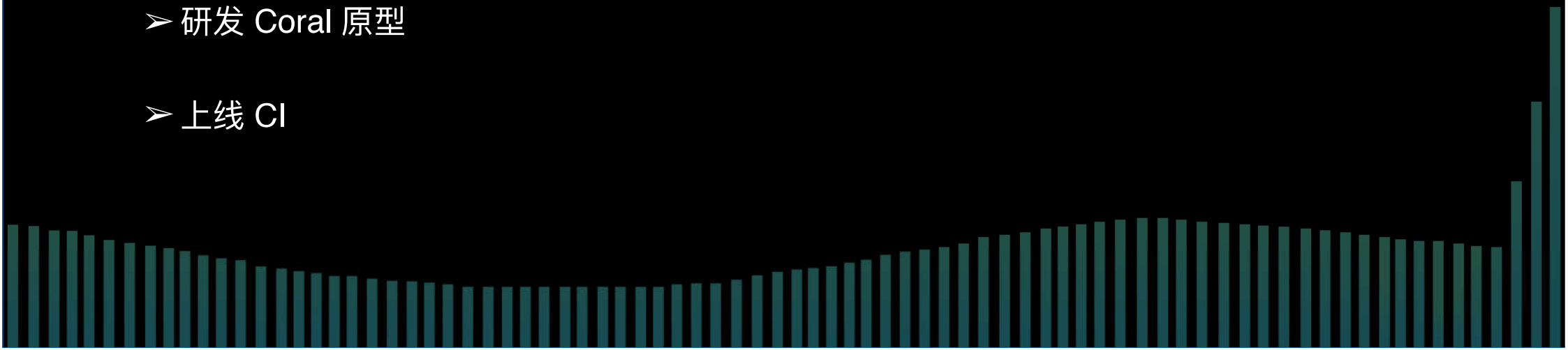
中兴契约测试工程管理实践

契约测试试点及推广，契约测试嵌入团队研发流程，契约测试成果和收益



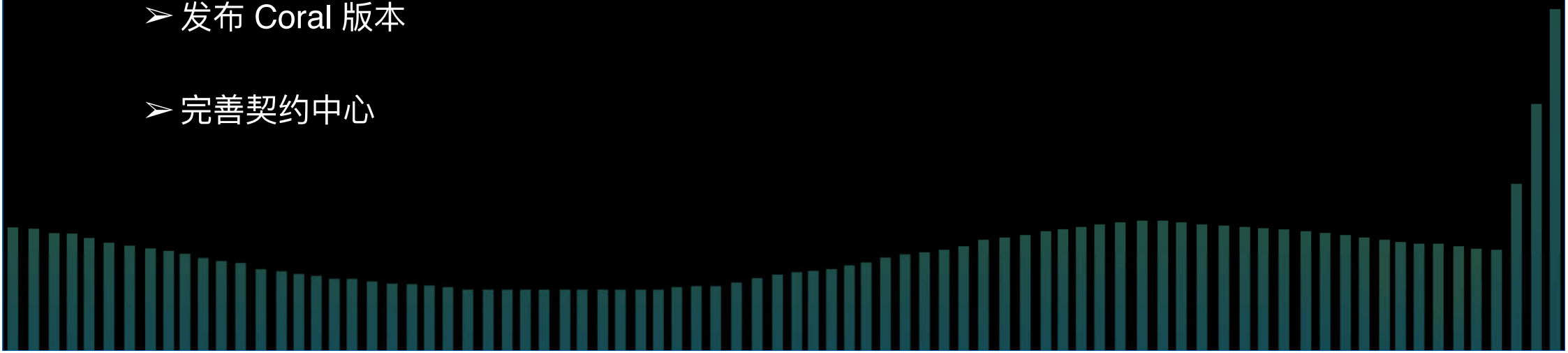
契约测试试点：自下而上

- 寻找游击队
- 确定试点团队
- 梳理典型用例
- 研发 Coral 原型
- 上线 CI

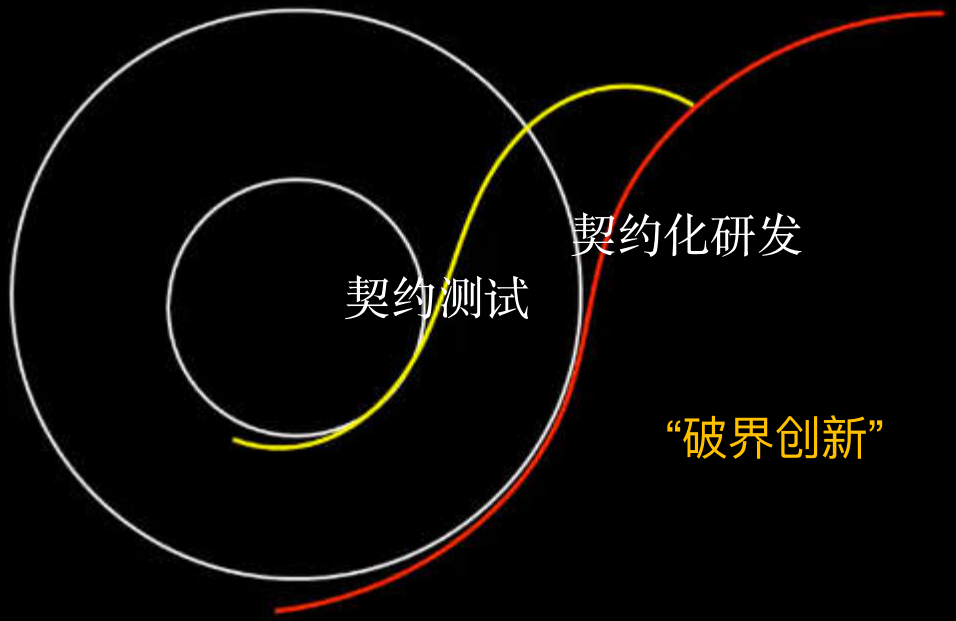


契约测试推广：自上而下

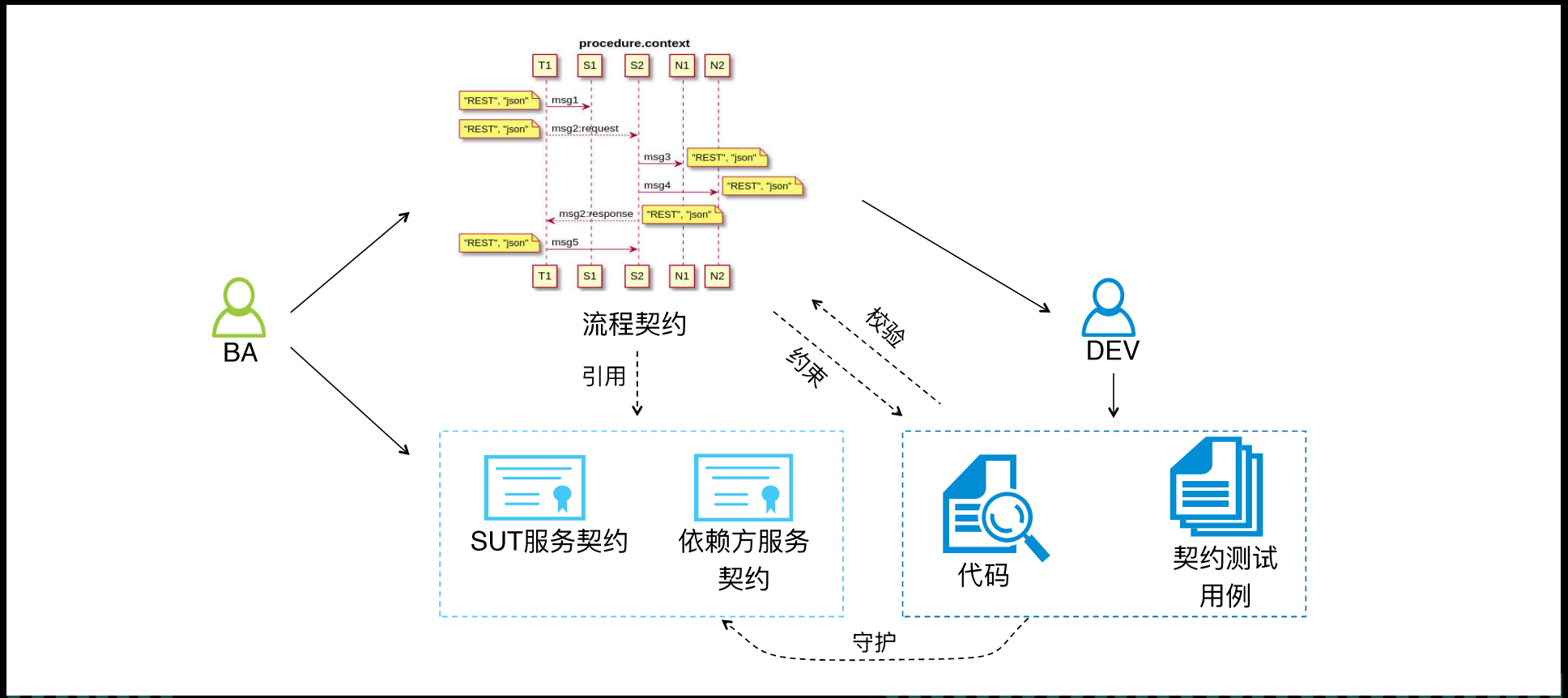
- 组建正规军
- 规划年度目标
- 多级任务管理
- 发布 Coral 版本
- 完善契约中心



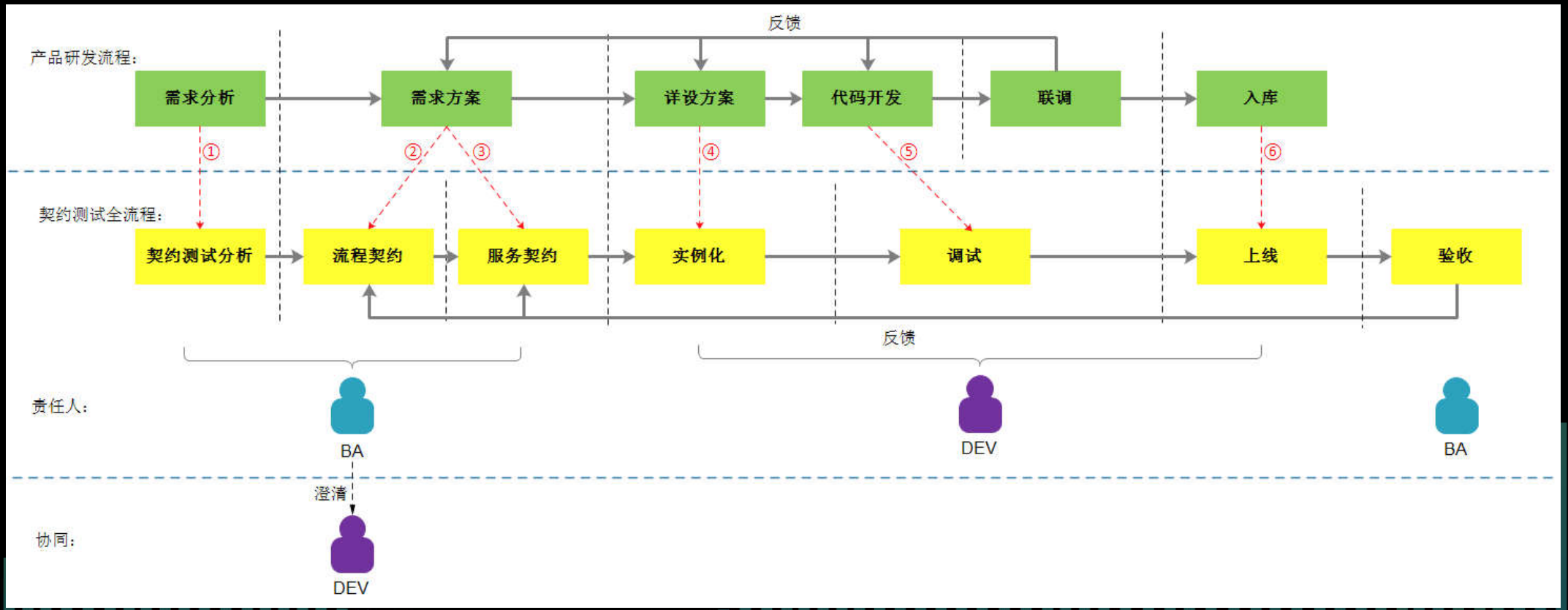
契约化研发



契约是 BA 和 DEV 的粘合剂



契约测试嵌入团队研发流程



成果

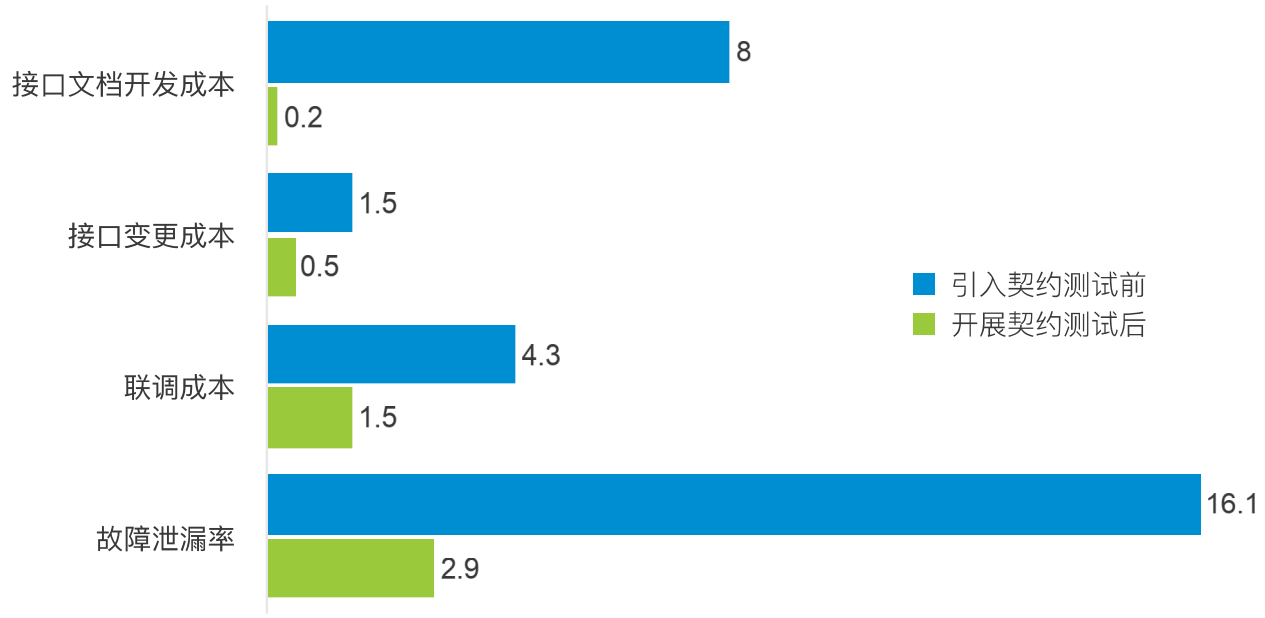
容器云平台项目

- 已上线服务 40 个
- 已完成用例 438 个，已上线用例 421 个

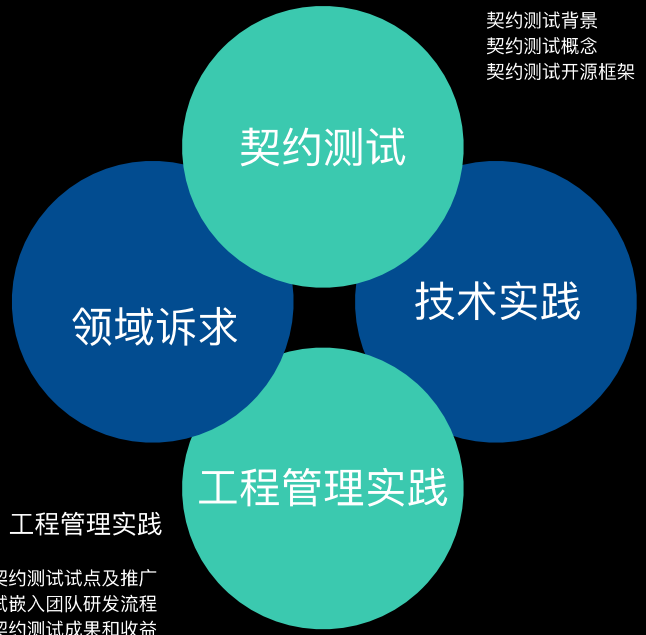
电信微服务平台项目

- 已上线服务 15 个
- 已完成用例 377 个，已上线用例 332 个

收益


接口文档开发成本
8
0.2
接口变更成本
1.5
0.5
联调成本
4.3
1.5
故障泄漏率
16.1
2.9
引入契约测试前
开展契约测试后

小结



契约测试

契约测试背景
契约测试概念
契约测试开源框架

领域诉求

SUT 基础设施尽可能真实
除过服务契约，还要有流程契约
无码化契约测试用例开发

技术实践

契约测试框架 Coral（自研）
契约测试用例开发套路
契约测试最佳实践

工程管理实践

契约测试试点及推广
契约测试嵌入团队研发流程
契约测试成果和收益

谢谢