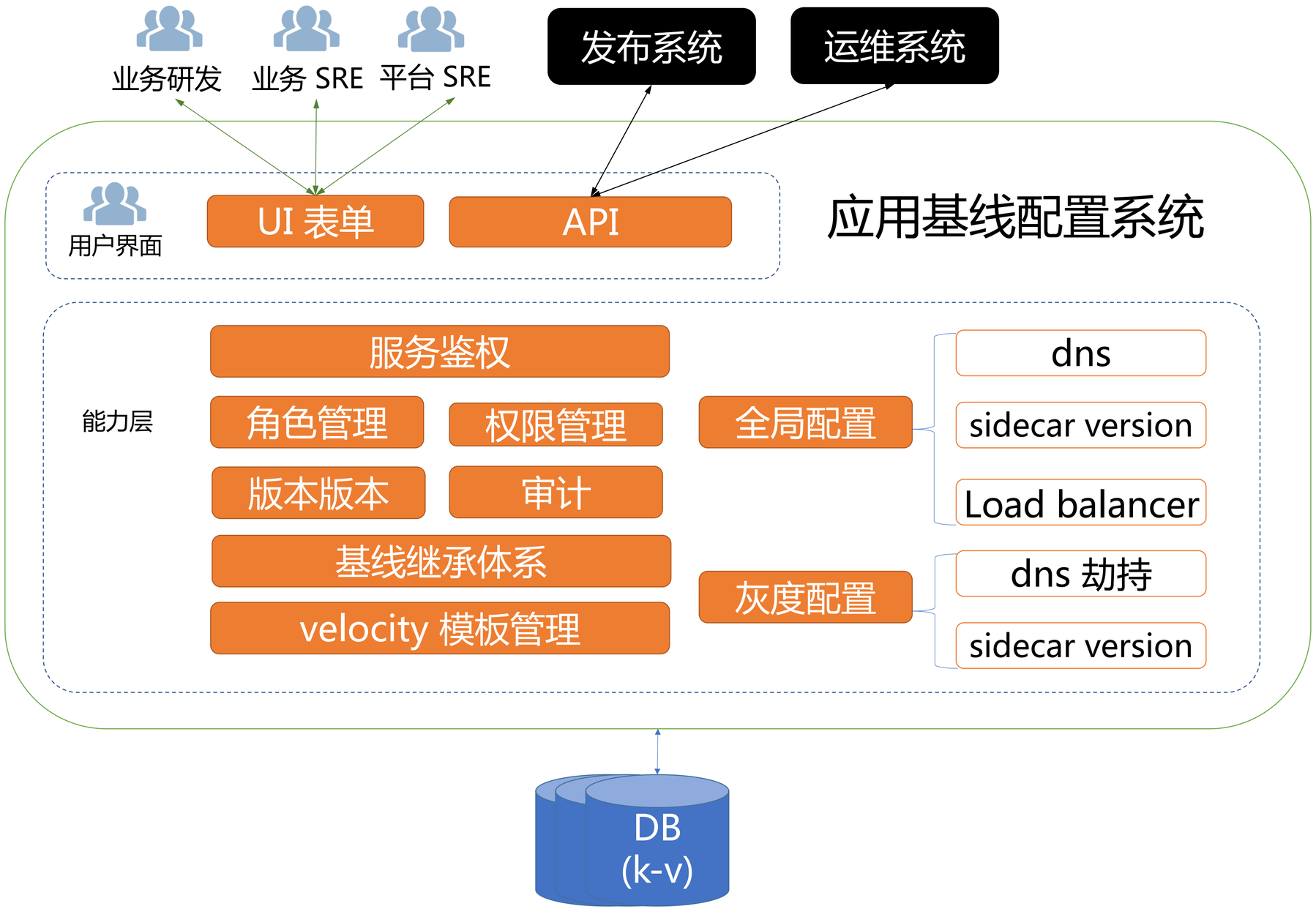
Kusion 在蚂蚁的规模化探索实践

莫城、半庭 | 部门描述或职位描述

蚂蚁集团应用基线系统现状

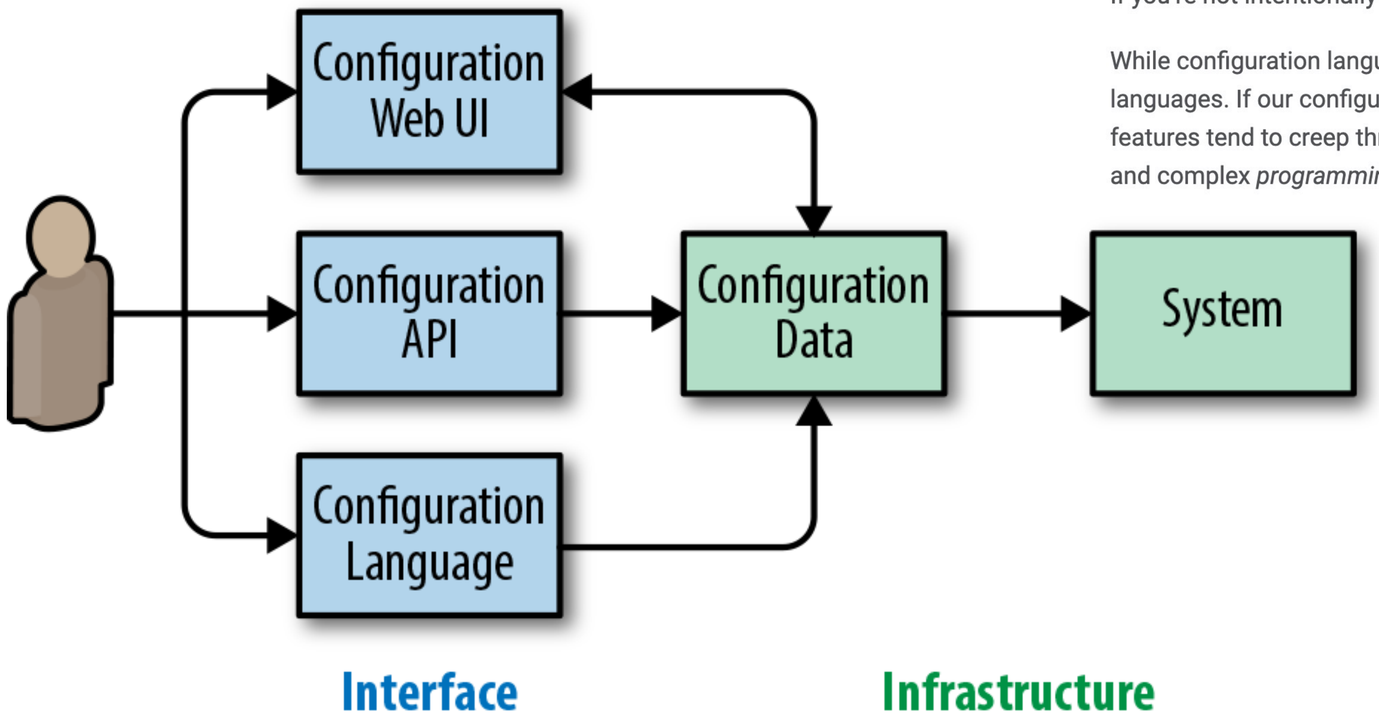


- BS 架构
- 模板化
- 配置继承

站在巨人的肩膀上

From Google SRE Book :

> 大规模配置管理的问题本质是语言问题

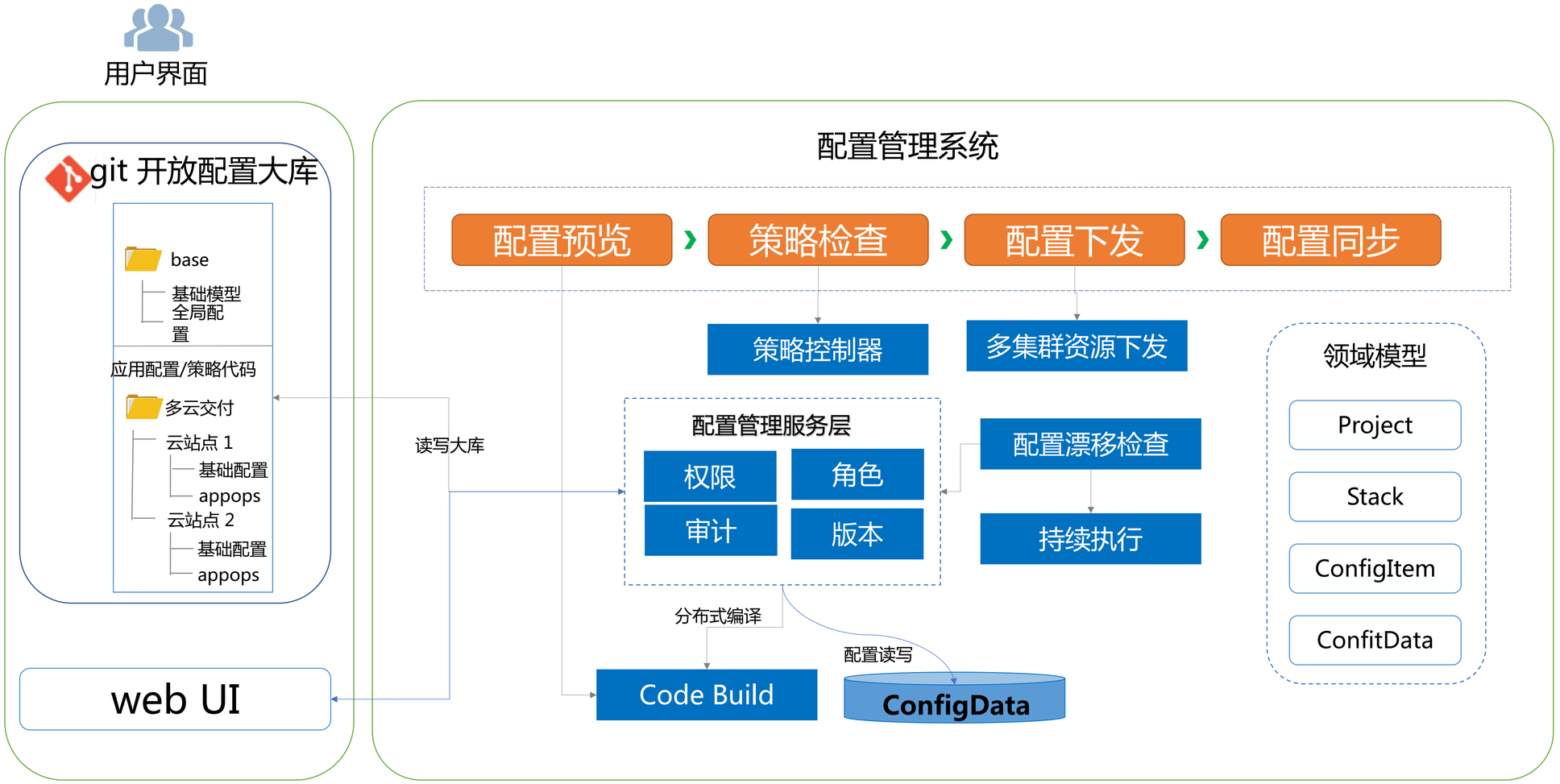


Pitfall 1: Failing to Recognize Configuration as a Programming Language Problem

If you're not intentionally designing a language, then it's highly unlikely the "language" you'll end up with is a good one.

While configuration languages describe data rather than behavior, they still have the other characteristics of programming languages. If our configuration strategy starts with the objective of using a data-only format, programming language features tend to creep through the back door. Rather than remaining a *data-only* language, the format becomes an esoteric and complex *programming* language.

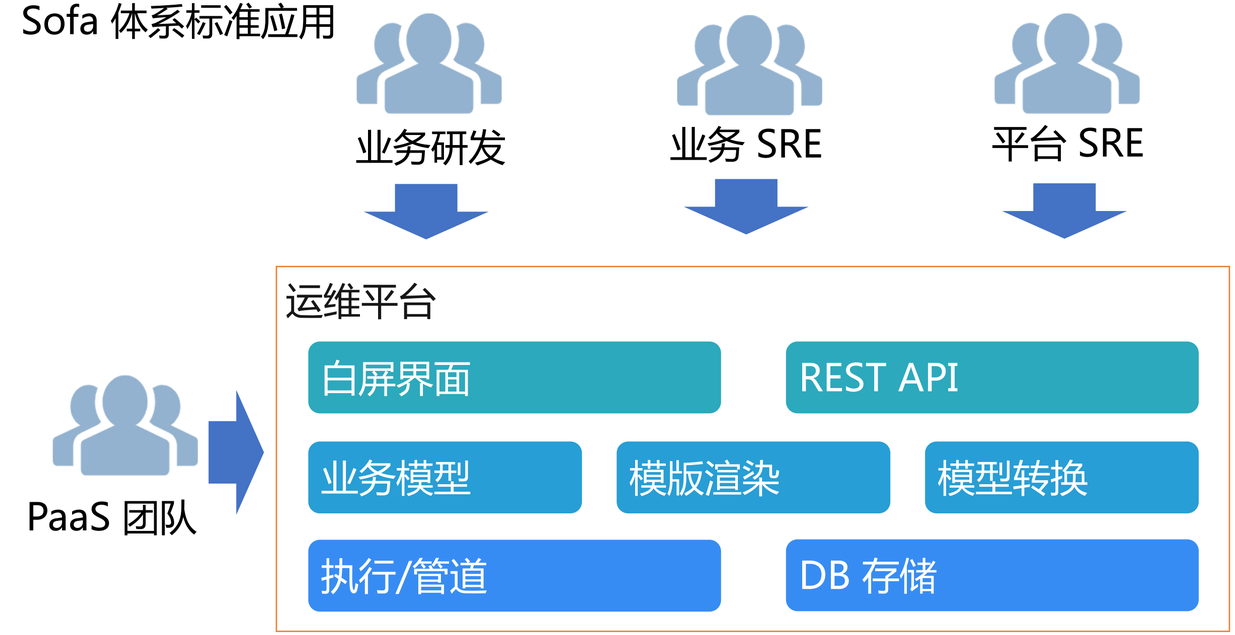
蚂蚁集团下一代配置管理系统



背景

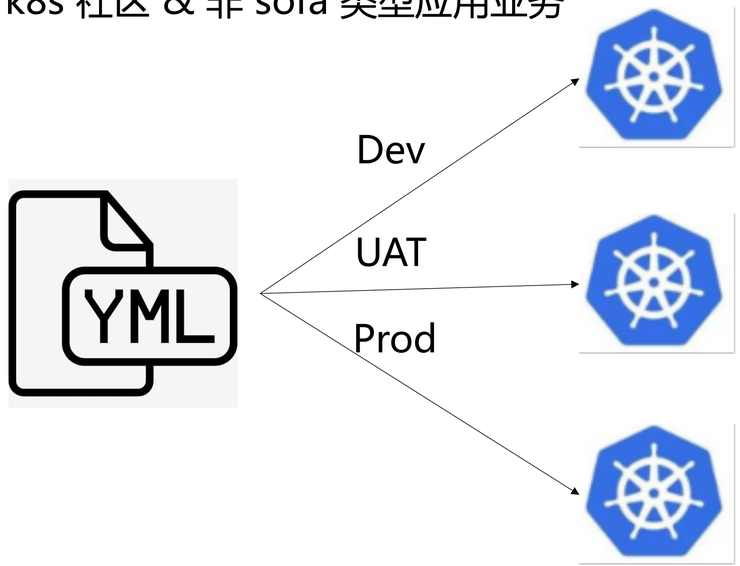
蚂蚁应用现状

Sofa 体系标准应用



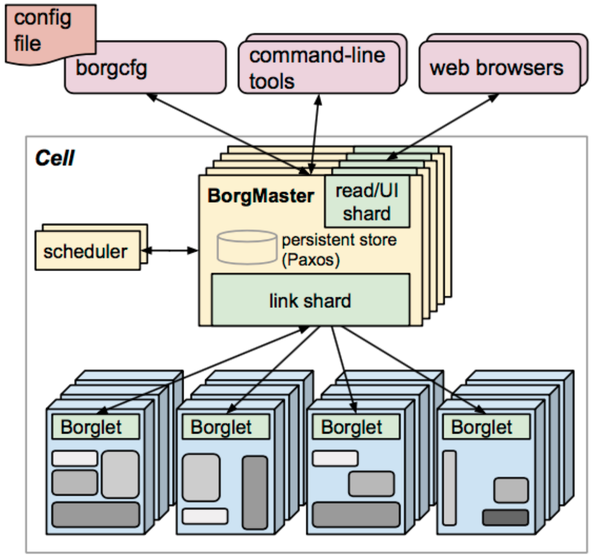
- 运维平台功能透明度不足，大量黑盒代码逻辑
- 运维平台高度定制化灵活性不足，无法满足应用个性化运维需求
- 依赖开发资源瓶颈，无法复用和快速实现SRE的独立业务需求

基于 k8s 社区 & 非 sofa 类型应用业务

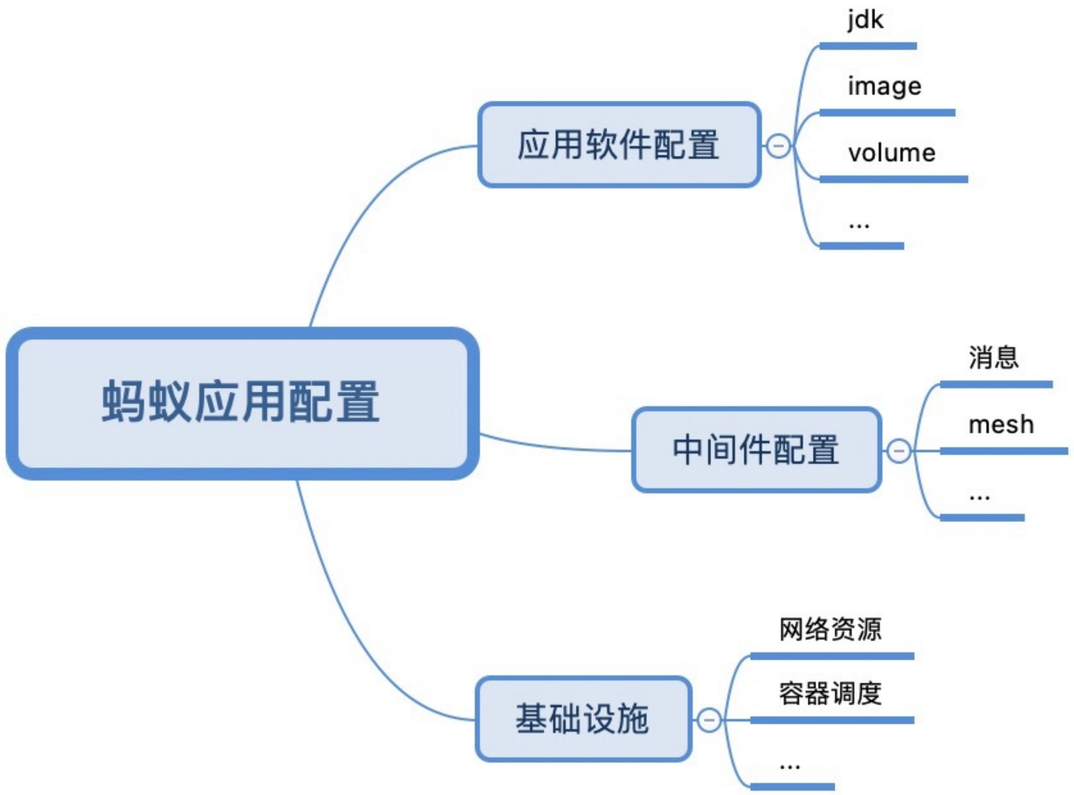


- 工程化水位低
- 无规划化流程，缺少
- 低协同性，大量文件拷贝

业界：谷歌 borg 大规模基础设施配置接入层管理 bcl , borgcfg, webconsole



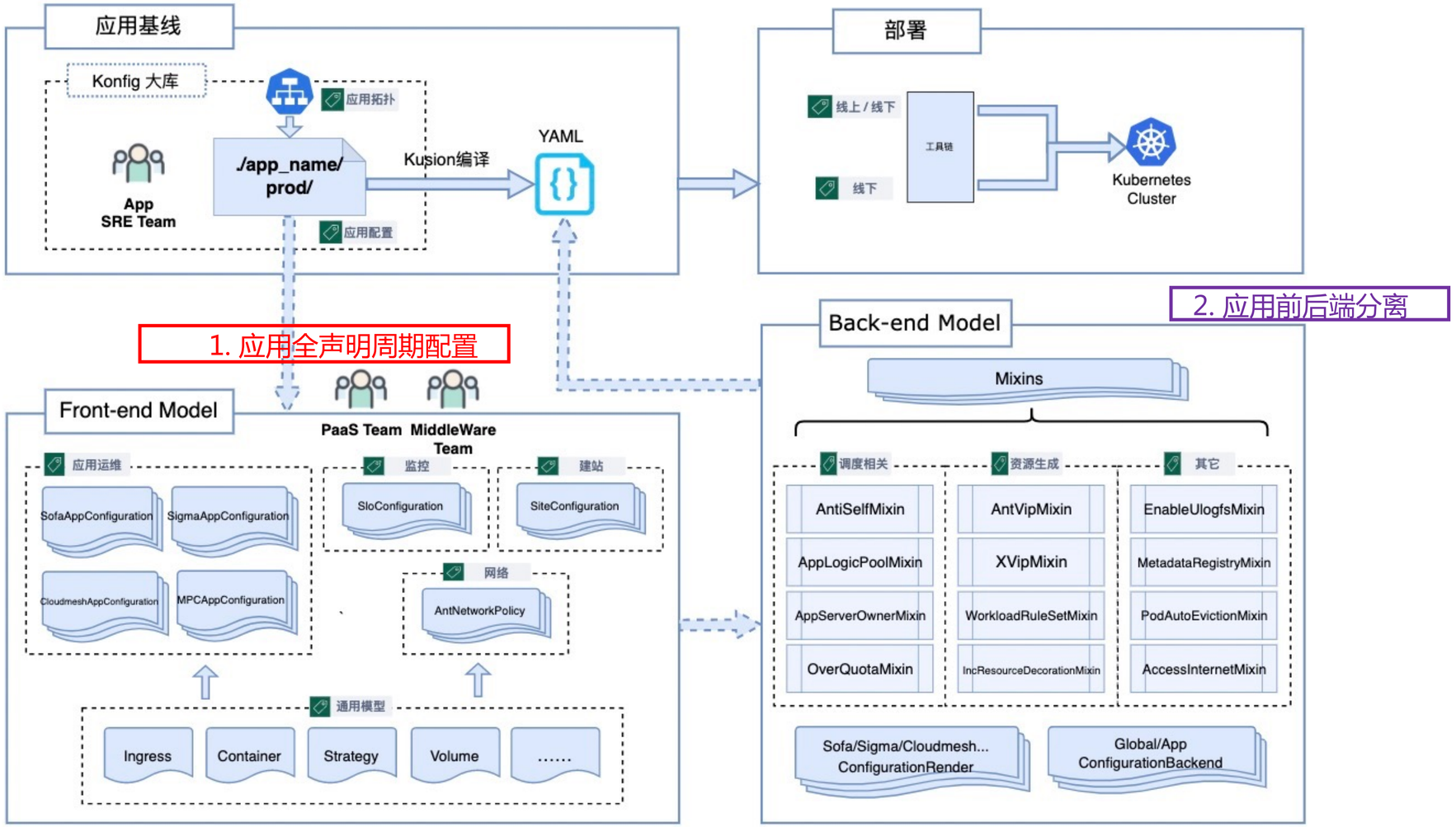

```
job hello_world = {
   runtime = { cell = 'ic' }              // Cell (cluster) to run in
   binary = '.../hello_world_webserver'   // Program to run
   args = { port = '%port%' }             // Command line parameters
   requirements = {     // Resource requirements (optional)
     ram = 100M
     disk = 100M
     cpu = 0.1
   }
   replicas = 10000    // Number of tasks
}
```



探索: 将蚂蚁应用依赖的基础设施配置-尝试探索通过类 bcl 的声明式模式，描述应用依赖的基础设施

蚂蚁声明式应用配置抽象

3. 配置收拢至统一大库 Konfig



4.多团队协同

案例-节点亲和性调度 Pod 实现

需求: 业务应用 Pod 不希望跟某些应用调度到同一 node

```yaml
spec:
  affinity:
    overrideAffinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
                - key: sigma.ali/app-name
                  operator: In
                  values:
                    - appname1
                    - appname2
```

K8s 定义亲和性定义

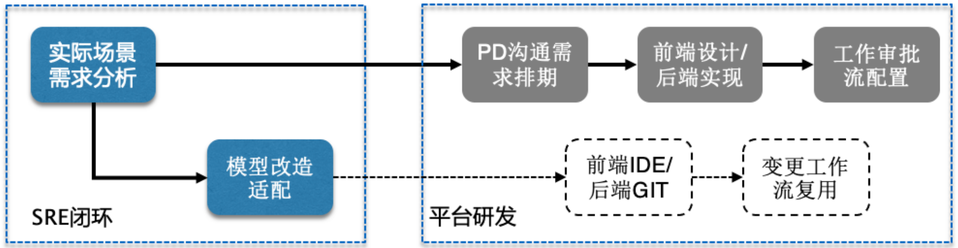
使用 kusion 编写后端实现模型

```
_appProhibitAntiAffinity?: [v1.PodAffinityTerm] = [{
    labelSelector: apis.LabelSelector{
        matchExpressions: [
            apis.LabelSelectorRequirement{
                key: "sigma.ali/app-name"
                operator: "In",
                values: data.prohibitAppList
            }
        ]
    }
    topologyKey: "kubernetes.io/hostname"
}] if "prohibitAppList" in data and data.prohibitAppList else Undefined
```

后端模型定义

后端模型定义具体实现逻辑

定义通用 appconfig 前端模型



1.本地编译查看生效结果
2.通过工具链渲染生效发布到线上

前端模型定义应用配置方法，简洁、干净

```
# 应用互斥配置  配置应用不允许调度到同一台node上
prohibitAppList?: [str]
```

业务 owner & 业务 SRE
定义应用配置

```
# 互斥应用列表
prohibitAppList = ["appname1", "appname2"]
```

跨团队模型分层协同

团队

应用研发/SRE 团队

应用研发/SRE 团队

PaaS 团队

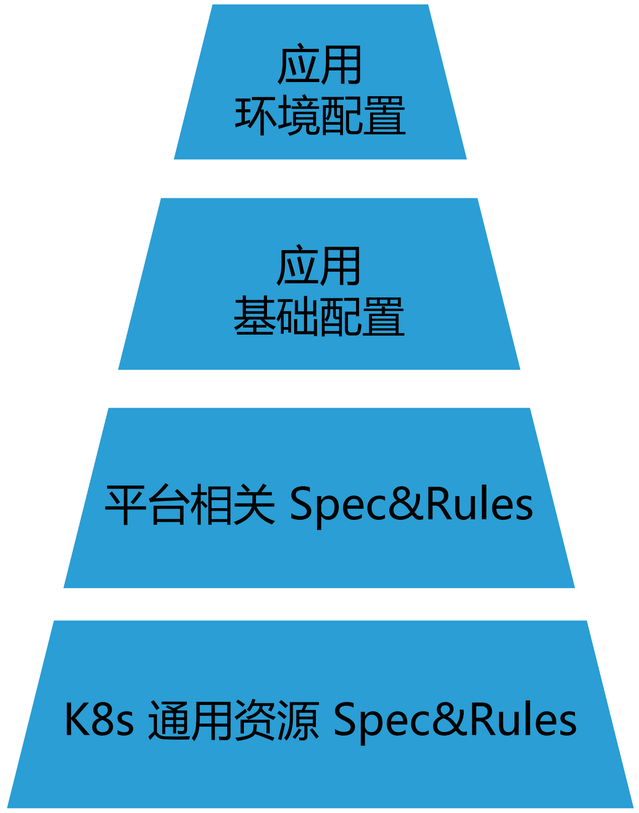
中间件团队

网络团队

调度团队

基础设施 SRE 团队

分层模型定义



Schema define, defaulting, validation, testing

关键代码示例

./project/prod/main.k

./project/base/base.k

sofa_app_conf.k

middleware.k

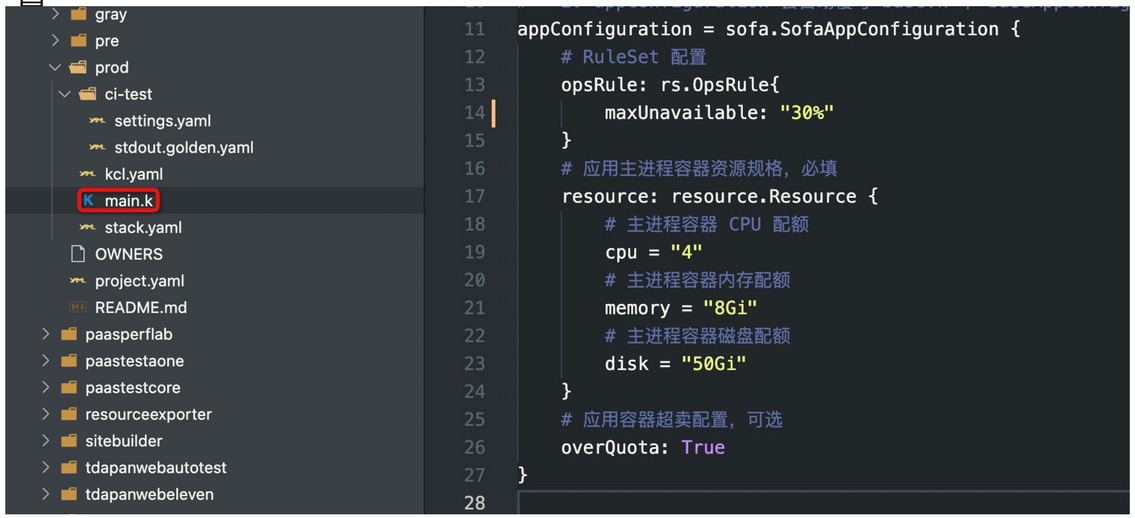
k8s_app_conf.k

rbac.k

案例-大库应用配置 & 本地渲染结果

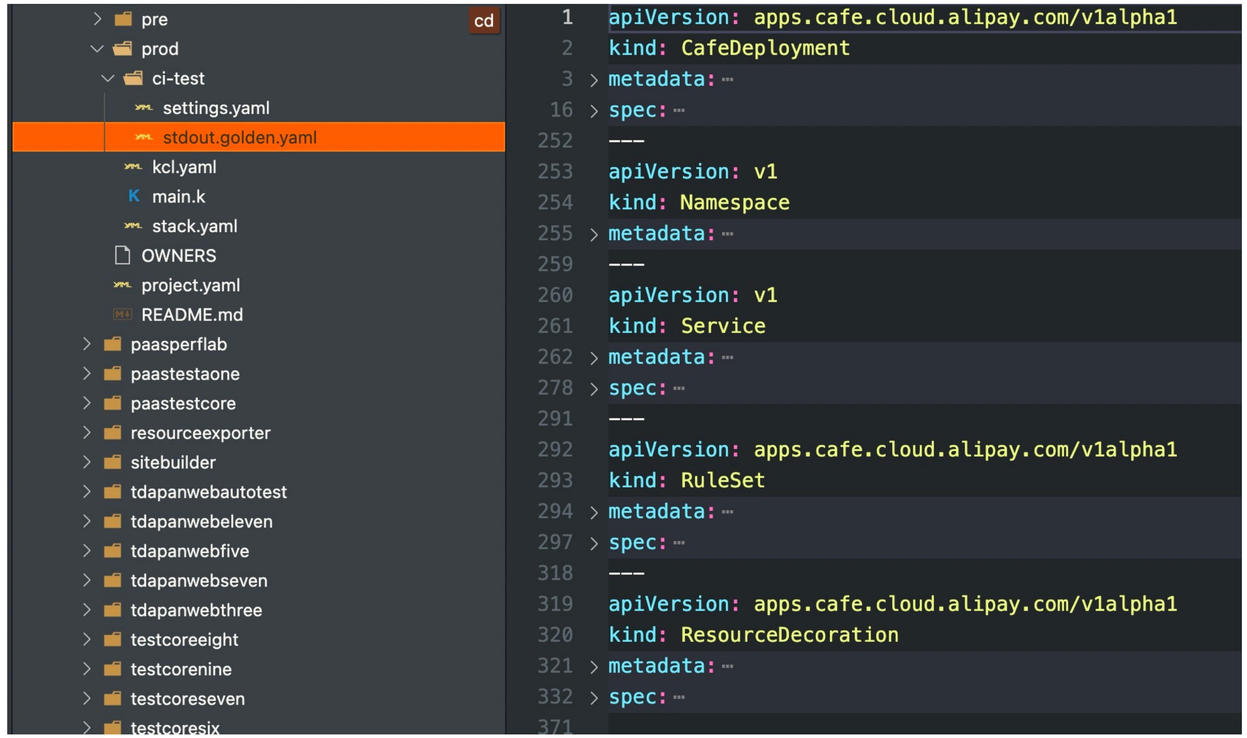
1.应用基础配置

2.应用环境配置

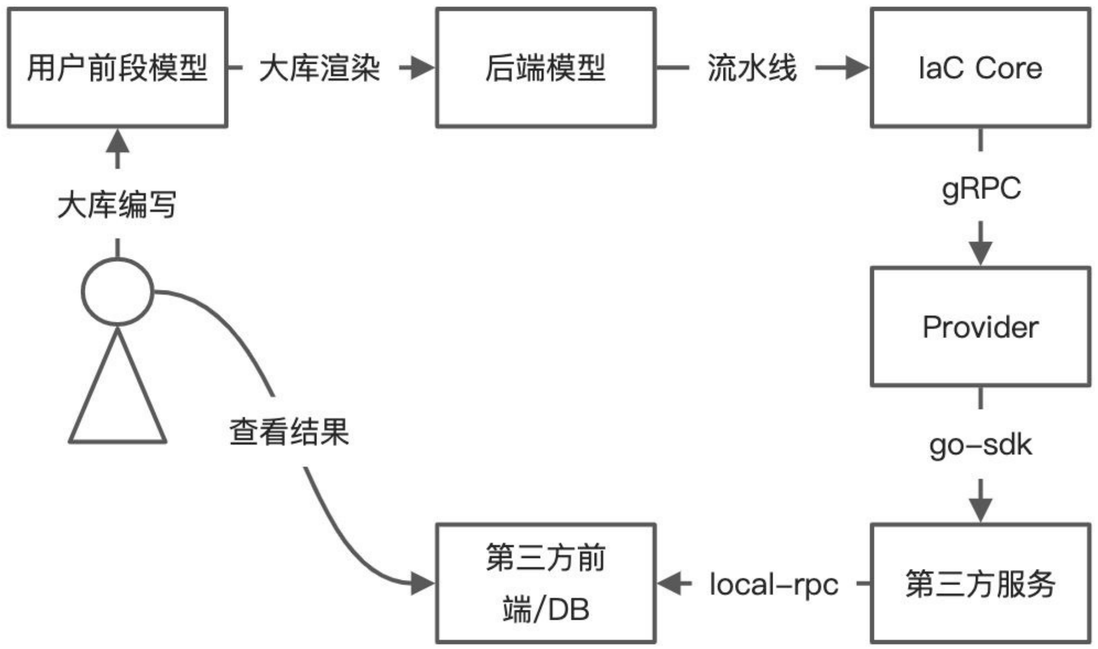


- 接入 xxx + 业务应用，基本覆盖蚂蚁所有技术栈类型
- 服务 xxx + 业务研发&SRE
- Xxw 次代码提交，xxxx + 发布分支及 Tag
- Xxxw 代码文件，xxw 行配置代码，xxxxw 行 YAML

3.应用资源本地渲染结果



探索-非 K8s 资源平台- iac 模式



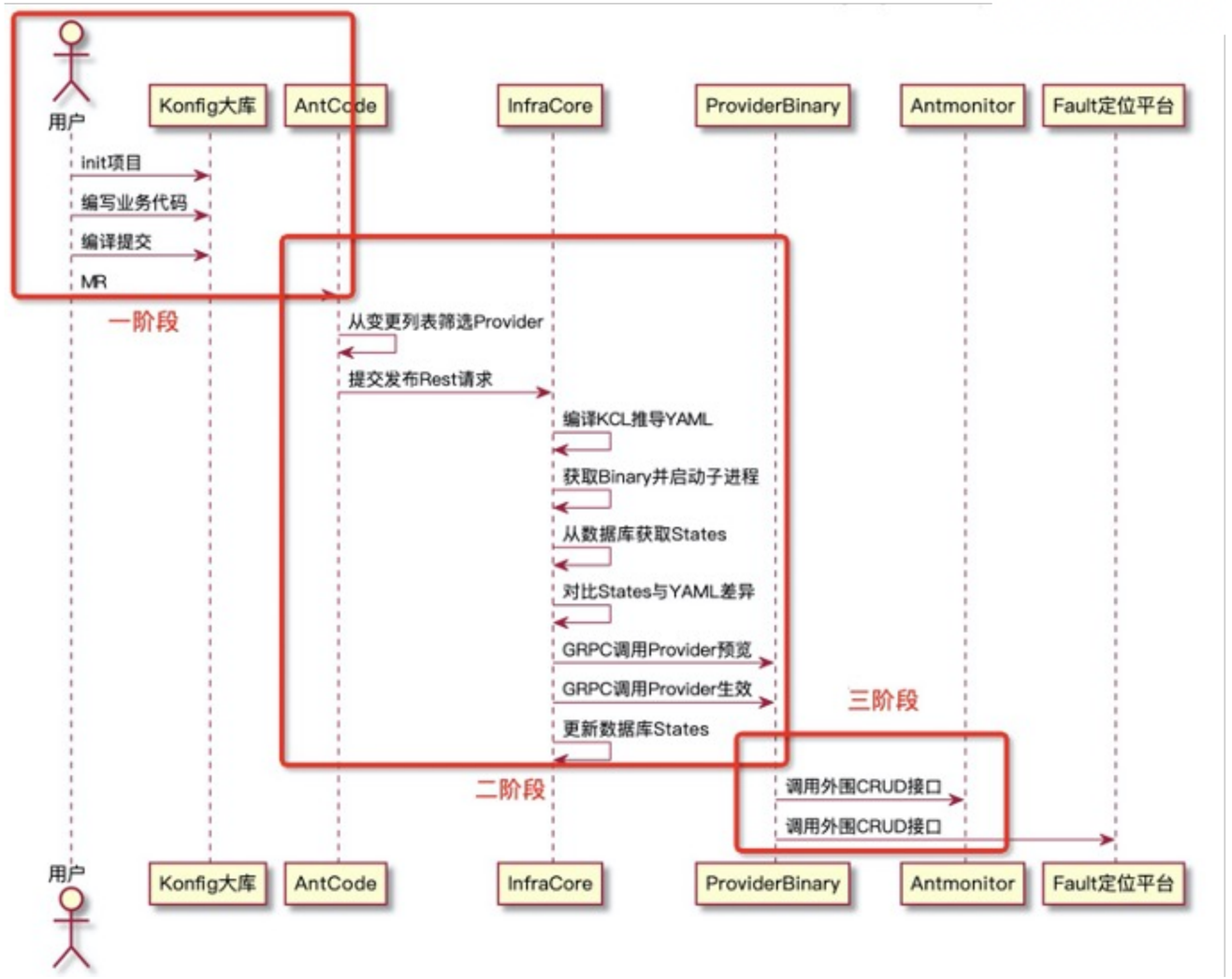

```
siteops.AppPlan {
    zone = __META_IDC_NAME
    apps = [
        # 创建机房级内网域名
        {
            appName = "meshpilot"
            postTask = [
                {
                    name = "dns_cname"
                    displayName = "创建机房级内网域名"
                    providerResource = {
                        command = "sitebuilder_dns_cname"
                        resource = _internetDomainCname
                    }
                }
            ]
        }
        # 发布 cafeextcontroller
        {
            appName = "cafeextcontroller"
            replicas = 3
        }
```

1.基础设施交付流程自定义编排

2.建站交付经验代码化

3.新技术支持交付效率

探索-技术风险能力 iac 模式描述



探索-配置管理方向的优化

待补充图

Q & A

欢迎大家用钉钉扫码
或钉钉搜索：42753001
加入 KusionStack 官方交流群

欢迎大家用微信扫码
加入 KusionStack 官方微信群

KUSION STACK

THANKS