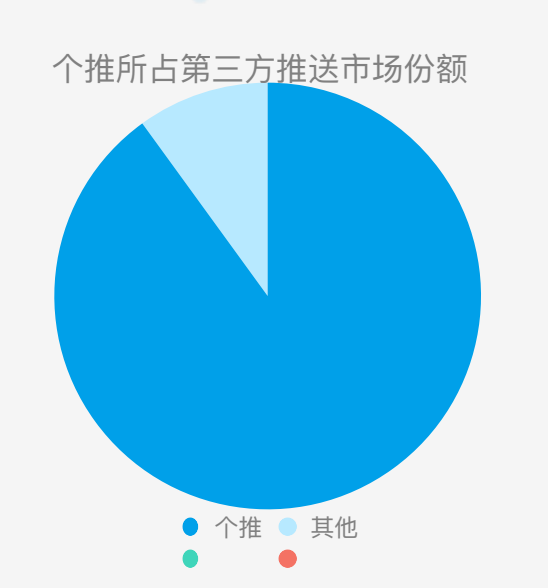

# 个推微服务实践

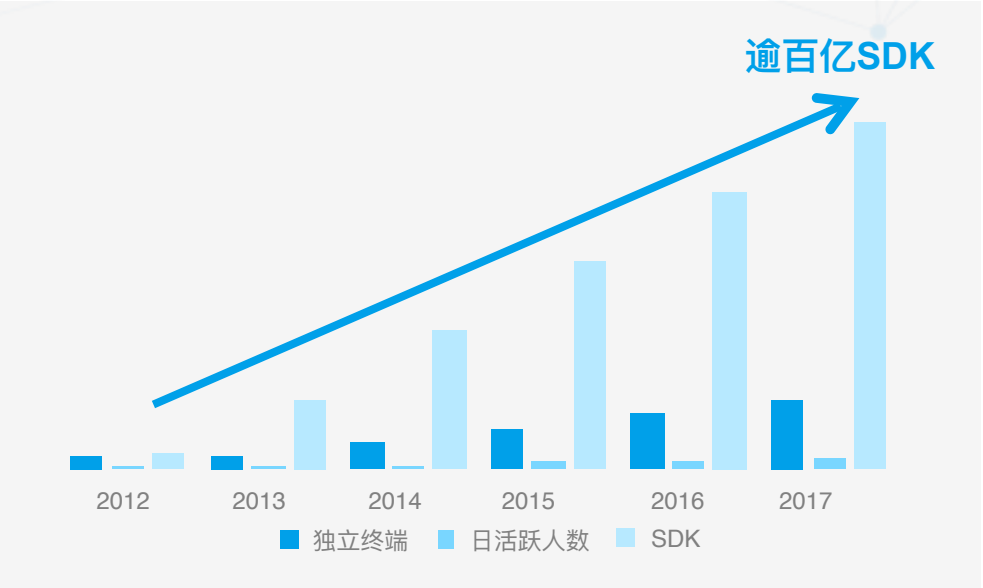
## 基于OpenResty和Node.js

平台研发总监 俞锋锋

2017-04-16

# 个推介绍




个推专注消息推送多年，拥有庞大的数据体系和深入的洞察能力。

个推 GeTui

# 什么是微服务

# 单体服务架构的缺点


单体服务架构，一个服务包，共享代码和数据。其缺点：

**开发成本高**

**可维护性差**

**技术选型困难**

**伸缩性差**

# 微服务架构的定义


微服务架构定义：

一个服务负责一项业务

服务独立部署

服务独立技术选型和开发

服务间松耦合

数据独立

# 微服务的优缺点

## 优点

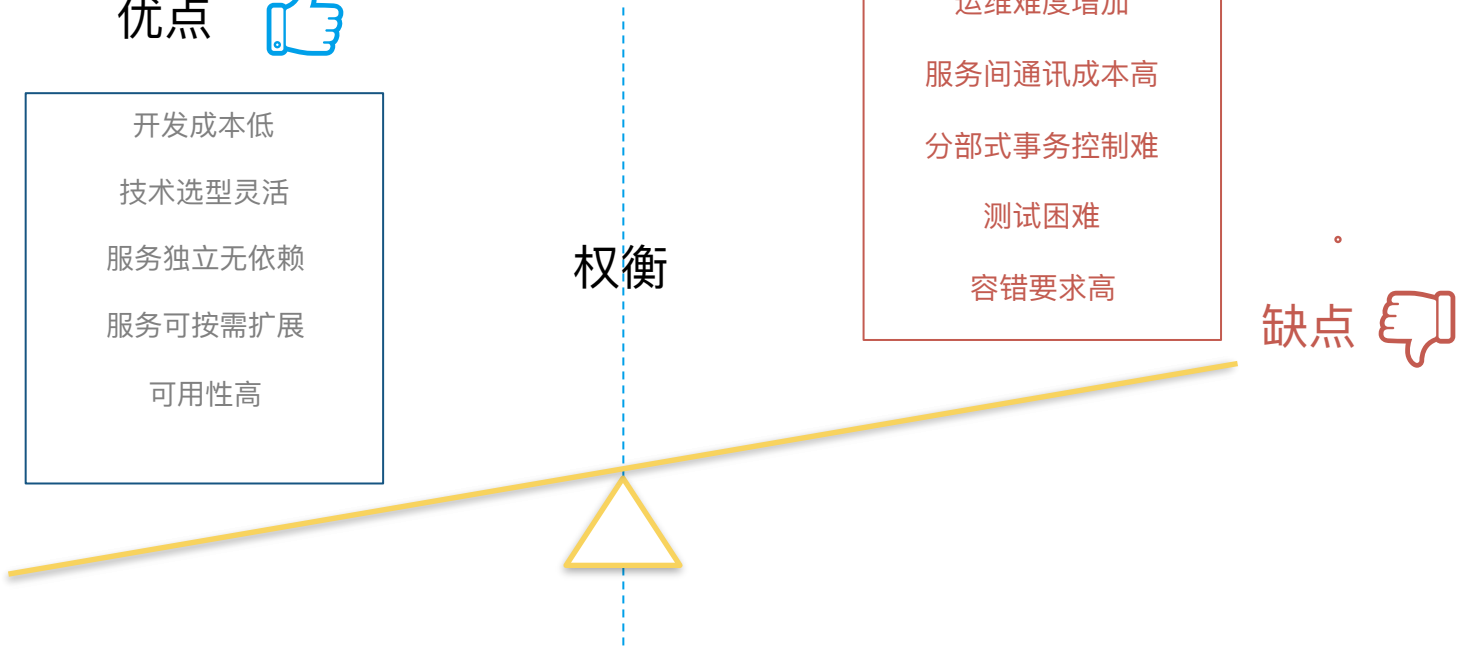
开发成本低

技术选型灵活

服务独立无依赖

服务可按需扩展

可用性高




## 缺点

部署复杂

运维难度增加

服务间通讯成本高

分部式事务控制难

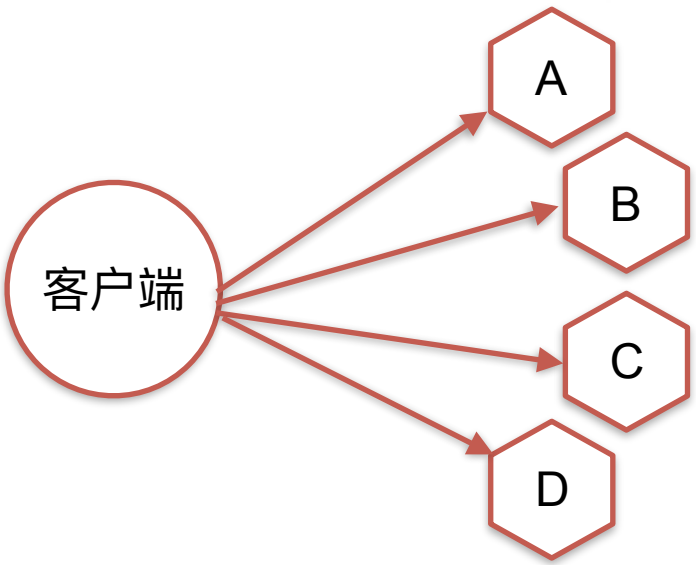
测试困难

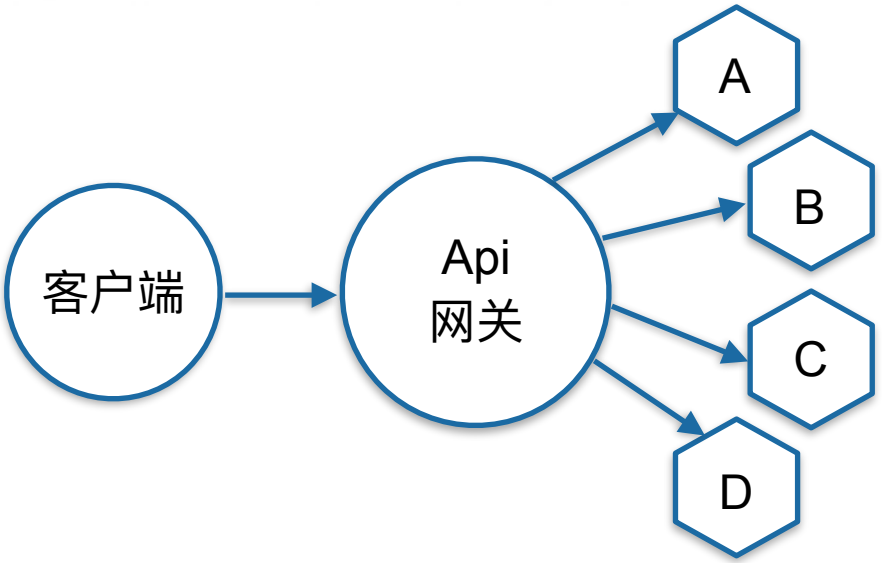
容错要求高

# 微服务的架构

# 微服务的架构

客户端和服务端的通信




多次服务请求，效率低

对外暴露服务接口

接口协议无法统一

客户端代码复杂，服务端升级困难

封装服务接口细节，减少通信次数

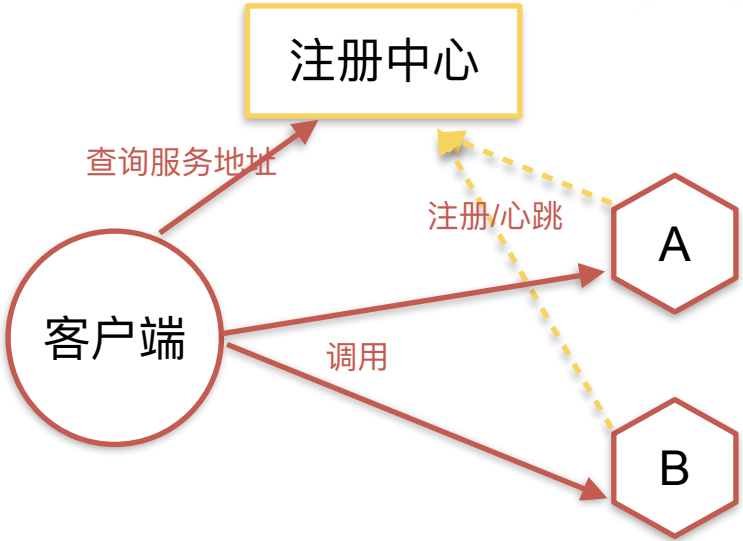
统一通信协议，减少客户端代码耦合

统一鉴权，流控，防攻击

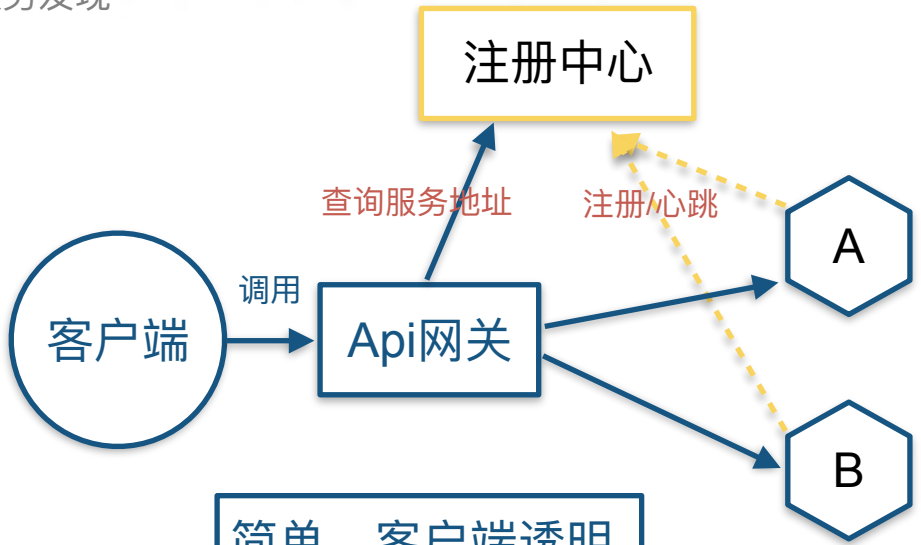
可能成为系统瓶颈

# 微服务的架构

## 服务发现

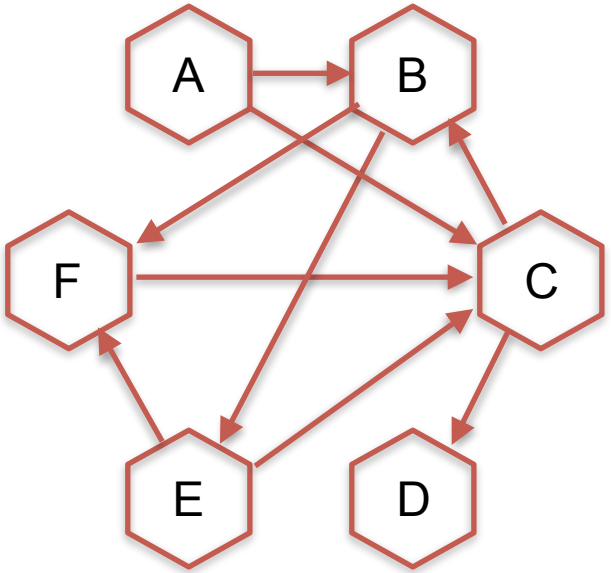



简单，扩展灵活
客户端维护服务端地址，耦合
开发成本高
协议升级难

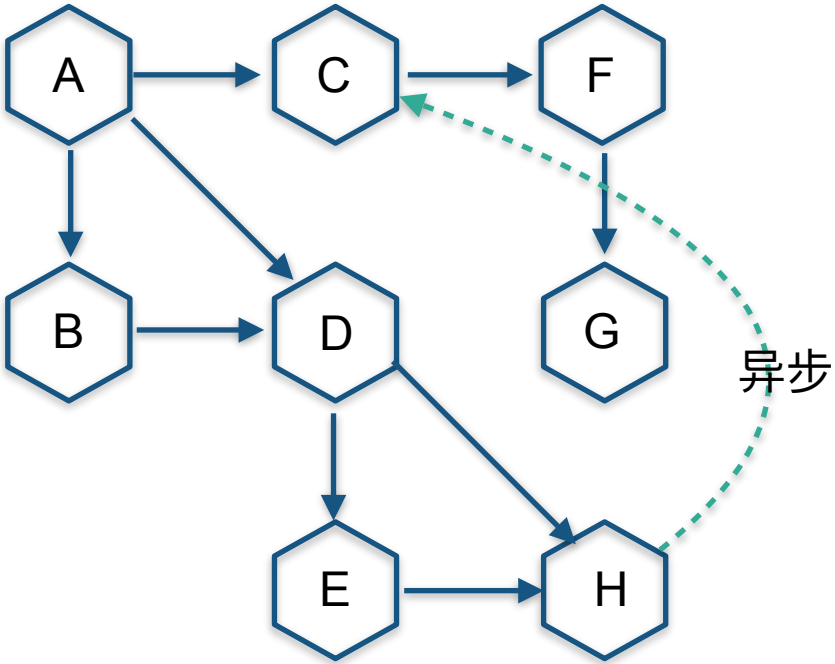



简单，客户端透明
统一鉴权，流控
需增加Api网关
Api网关高可用
额外延时

架构设计

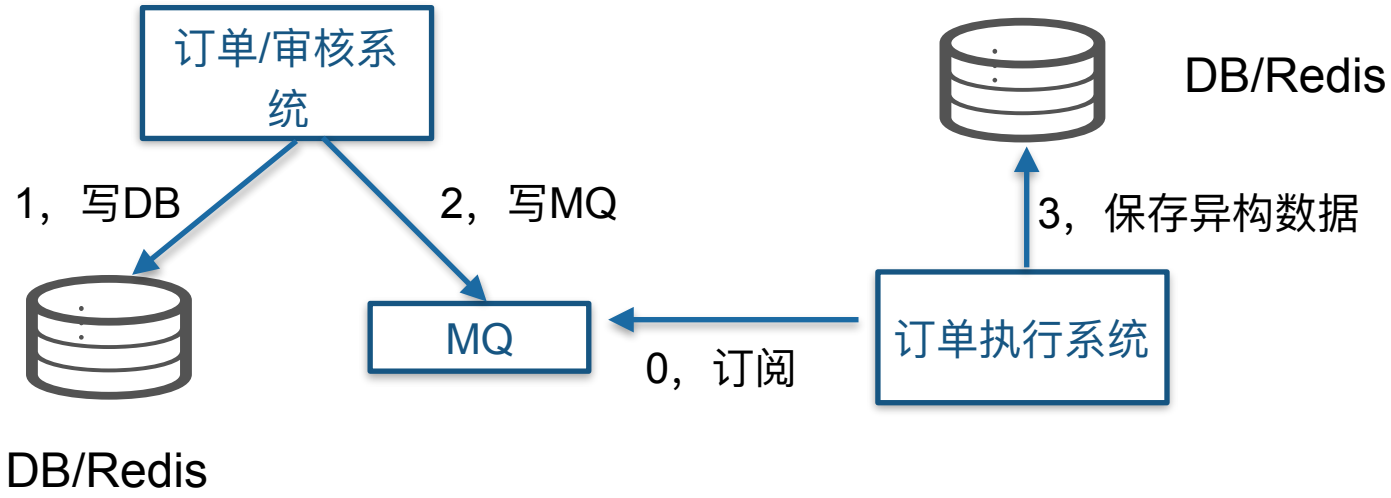



网状结构




分层结构

## 消息队列




利用消息队列进行异步解耦

# 微服务的架构

## 架构设计

合理拆分力度

尽量独立，避免暴露细节

服务要有抽象

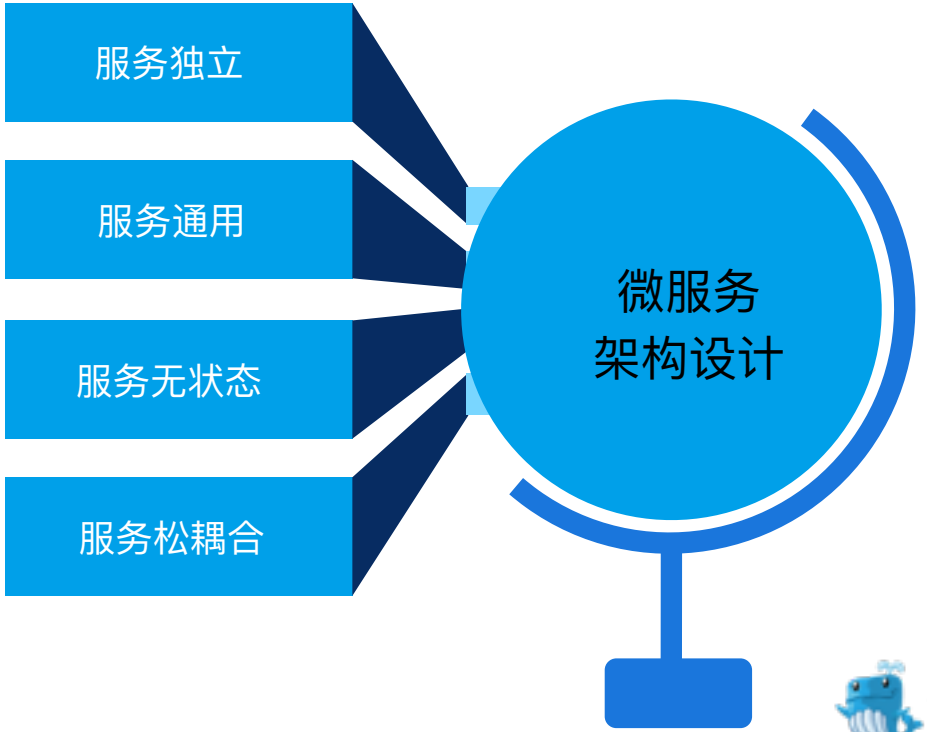
避免和业务强耦合

服务需无状态

或通过异步方式解耦

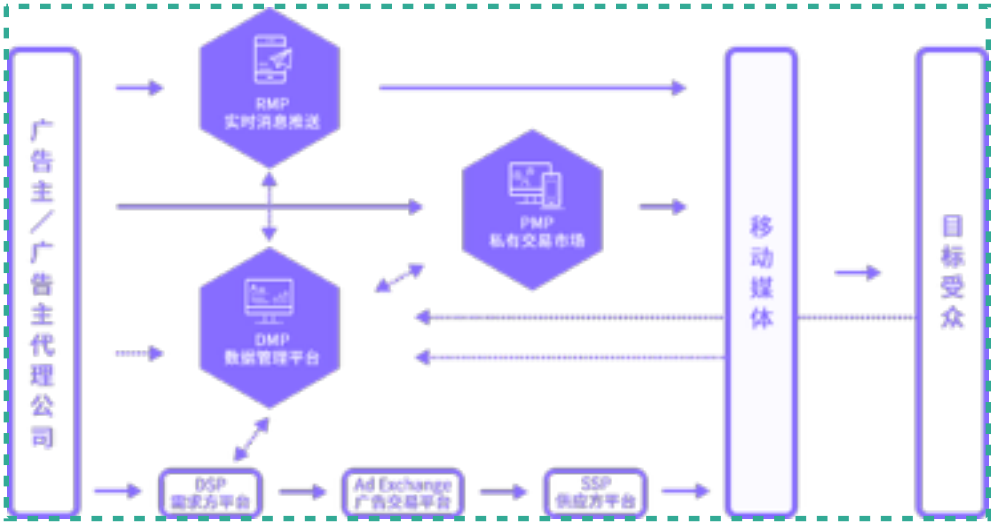
减少依赖

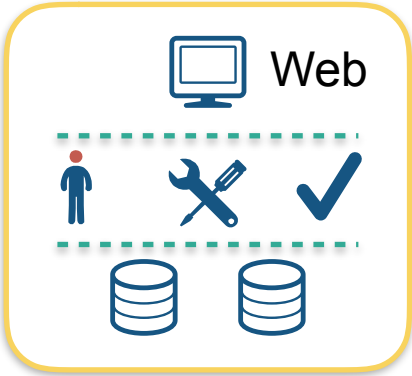
强调服务层次



# 个推微服务实践

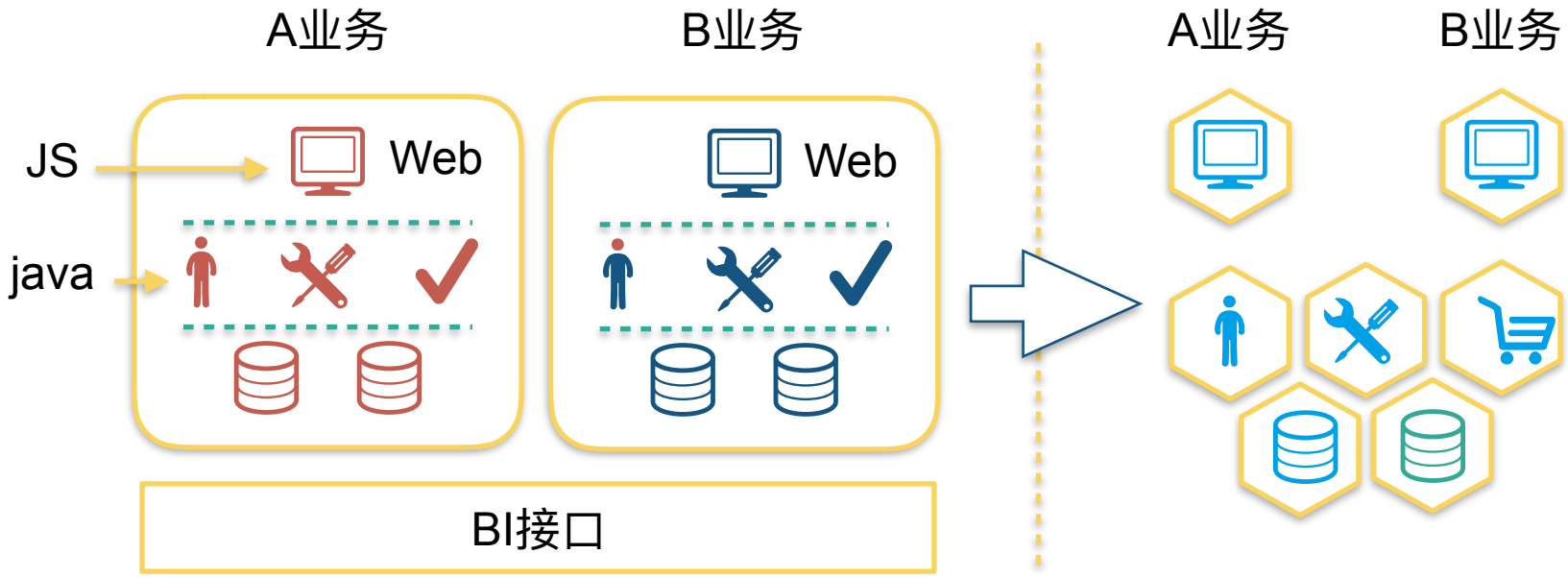
## 个推服务的三种场景


个推 GeTui.com

## 要解决的问题



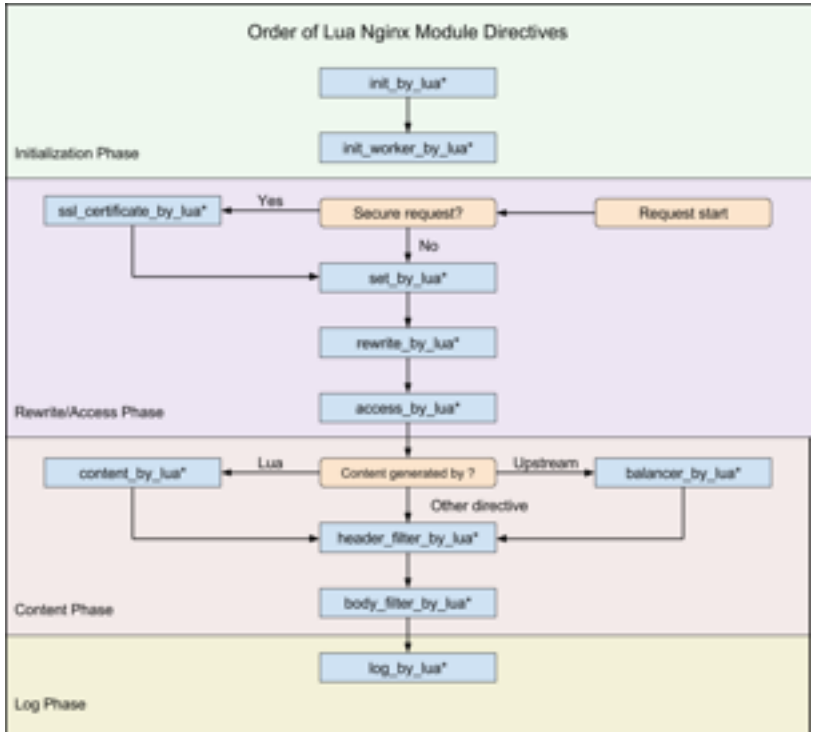

微服务，轻量级
应用和运行平台分离，便于运维
高并发，异步非阻塞

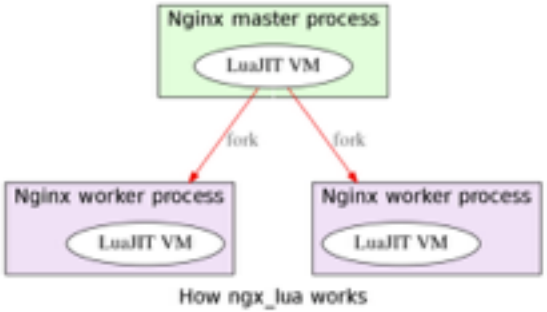
技术选型

| | 选型 | 特点 |
|---|---|---|
| Api网关 | OpenResty（Nginx+Lua） | 基于Nginx，扩展了异步非阻塞的Lua脚本支持，可构建超高并发web服务 |
| 客户端通信 | HTTP REST，JSON | 简单，可扩展，易开发，测试友好 |
| 服务间通信 | HTTP REST，JSON，消息队列 | 简单，可扩展，通过消息队列解耦 |
| 服务发现 | ZK，服务端发现 | 统一处理鉴权，流控，防攻击 |
| 语言选择 | Lua，Node.js，Java | 优选异步非阻塞的脚本语言，上手快，开发成本低，轻松实现高并发。支持Java实现服务容器，框架和语言无关 |

# OpenResty

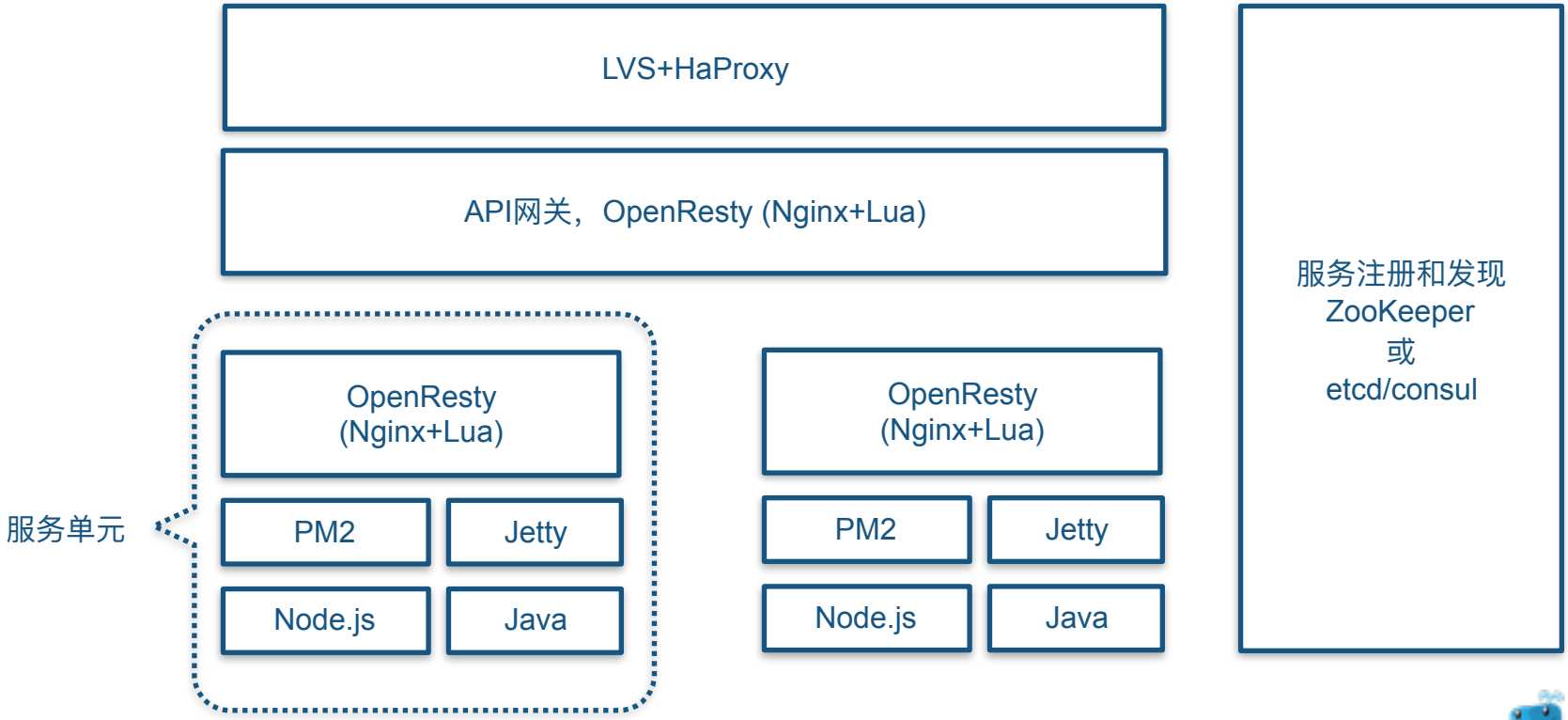
OpenResty是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。用于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。



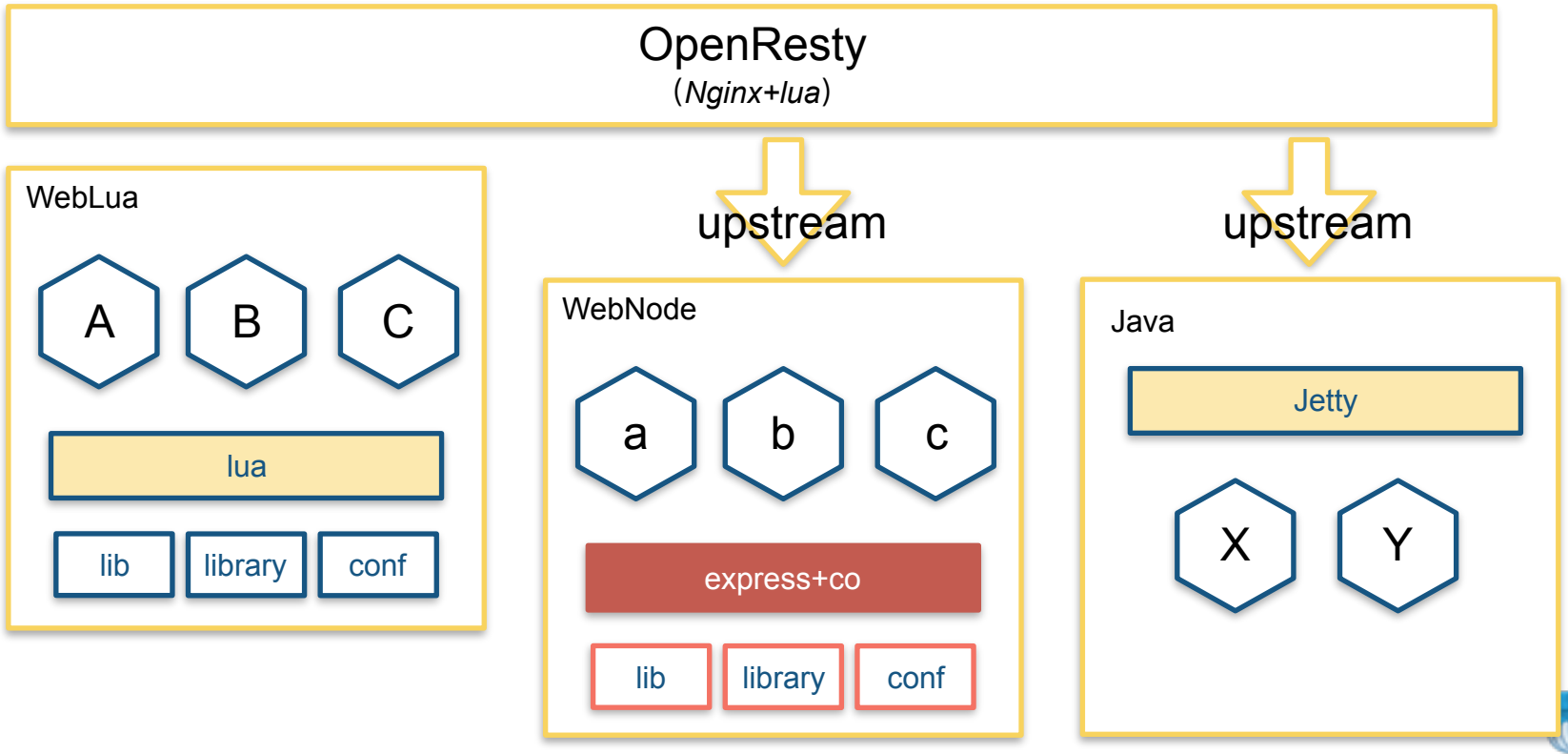




How ngx_lua works

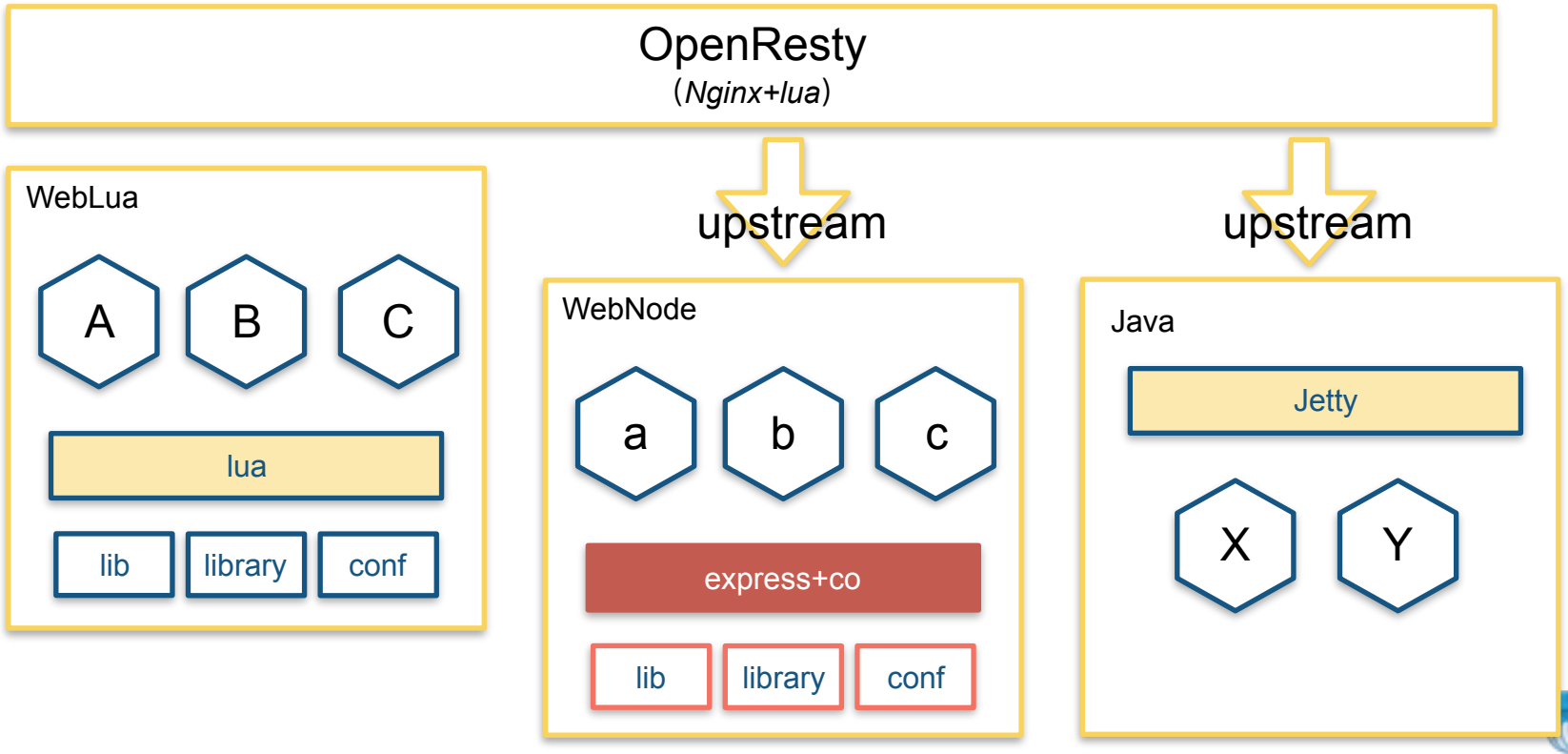
## 整体架构



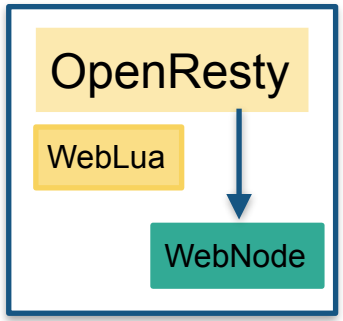
## 统一服务框架-服务单元



## 统一服务框架-服务单元



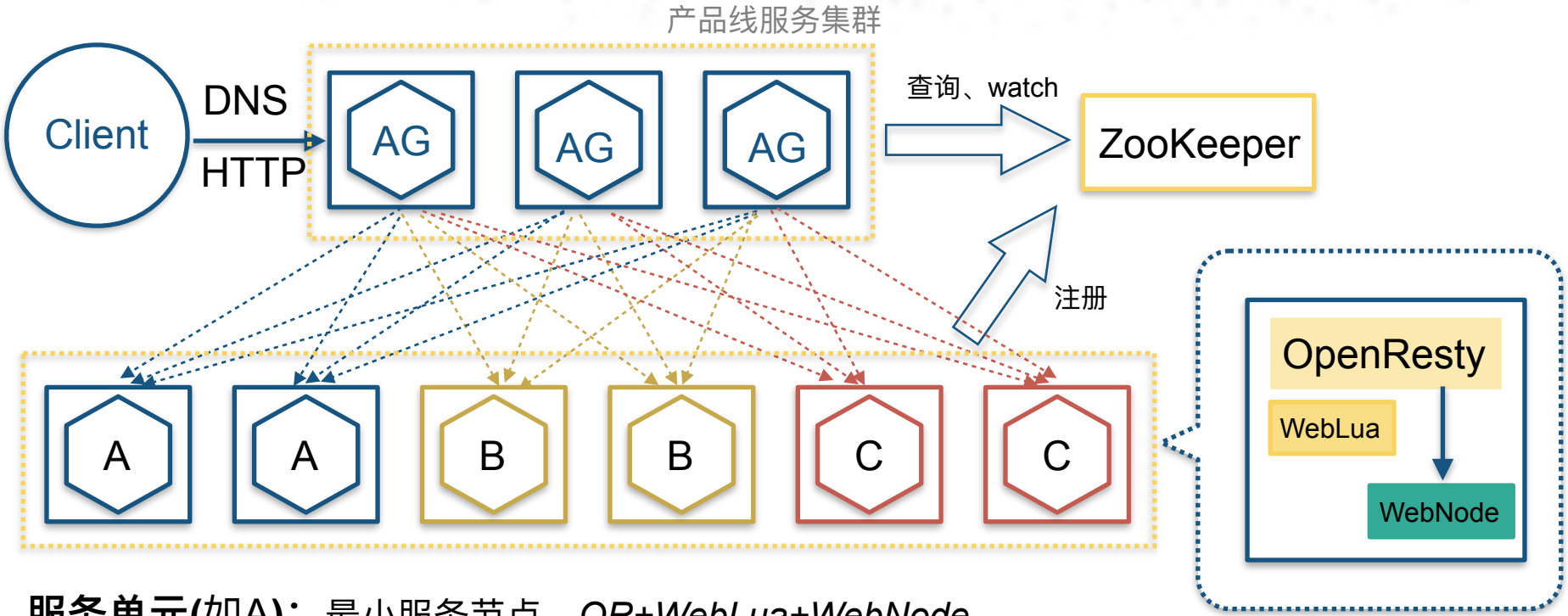
## 服务接入




```
upstream node {
    server 127.0.0.1:3000;
}

server {
    listen       6080;
    server_name  localhost *.a.com *.b.com *.c.com;

    location ^~/auth {
        rewrite ^/(\w+)/(\w+)/(\w+)?(.*)$ /?app=$1&action=$2&func=$3&$4 break;
        content_by_lua_file weblua/lua/content/index.lua;
    }

    location /tdesk {
        rewrite ^/(\w+)/(\w+)/(\w+)?(.*)$ /api/?app=$1&action=$2&func=$3&$4 break;
        access_by_lua_file weblua/lua/access/index.lua;
        proxy_pass http://node;
    }

     location /demo {
        rewrite ^/(\w+)/(\w+)/(\w+)?(.*)$ /api/?app=$1&action=$2&func=$3&$4 break;
        access_by_lua_file weblua/lua/access/index.lua;
        proxy_pass http://node;
    }
}
```

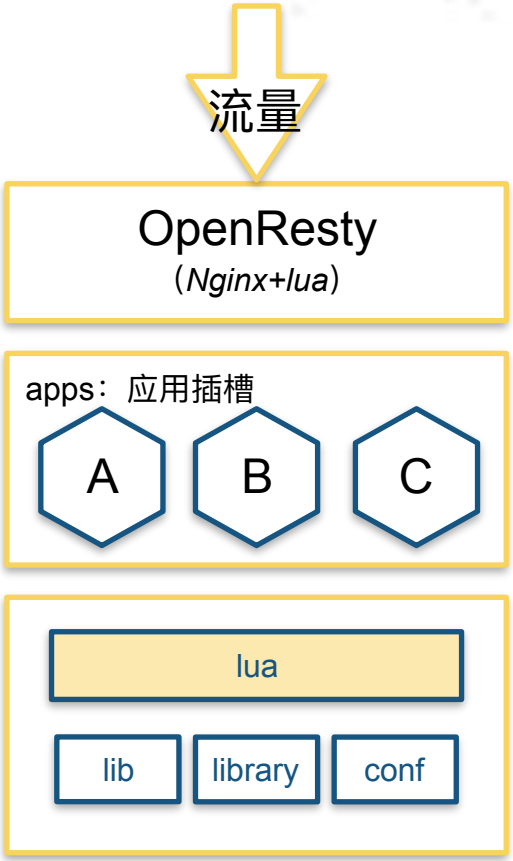
**产品线接入：**
*server*
**产品接入：**
*location*
**统一鉴权：**
*access_by_lua*
**Node服务：**
*proxy_pass*
*upstream*

个推 GeTui.com

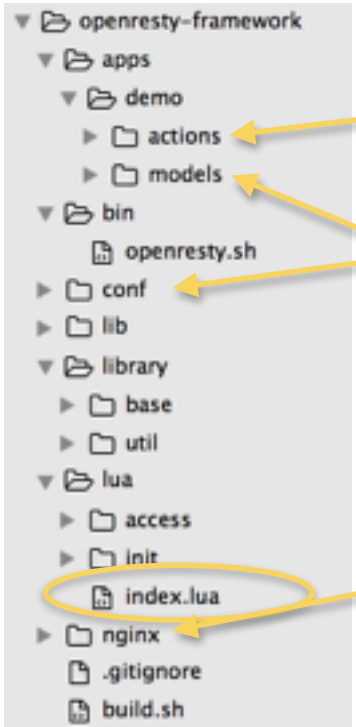



**服务单元(如A)：** 最小服务节点，*OR+WebLua+WebNode*

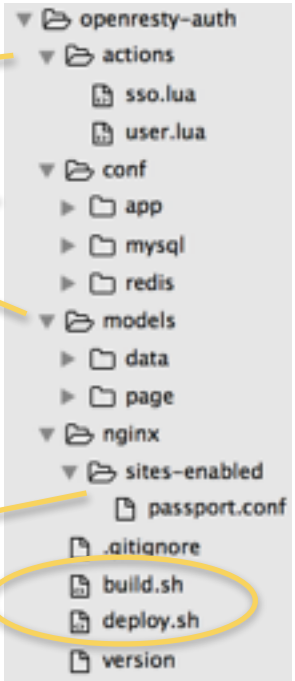
**AG：** Api网关，提供路由，流控，防攻击等




WebLua

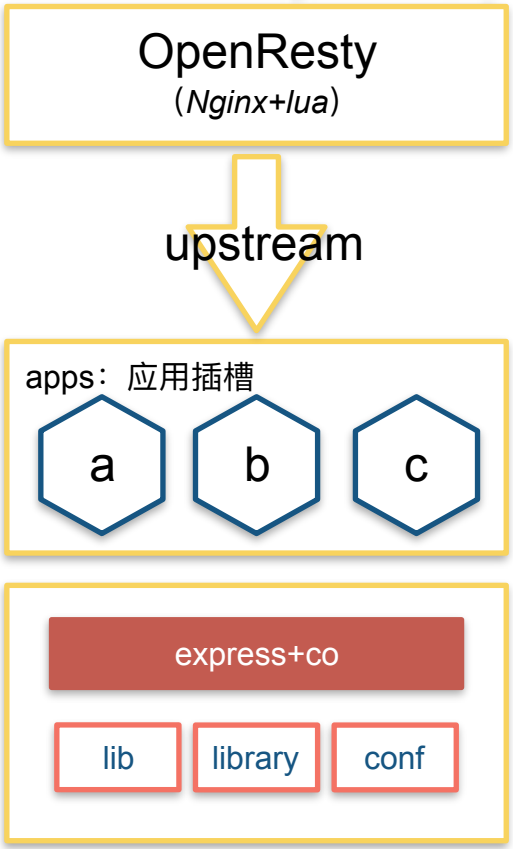



Build&
Deploy

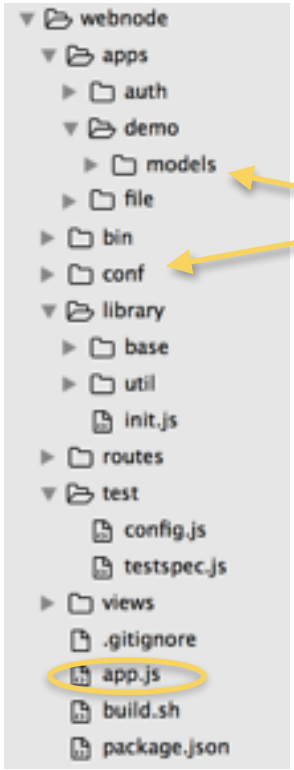



在线运行平台

Git代码

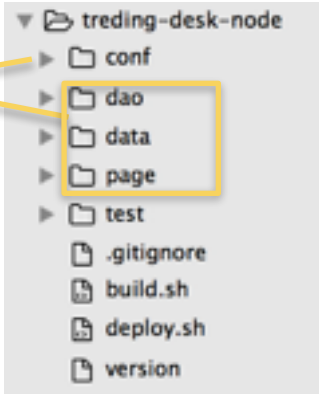



WebNode



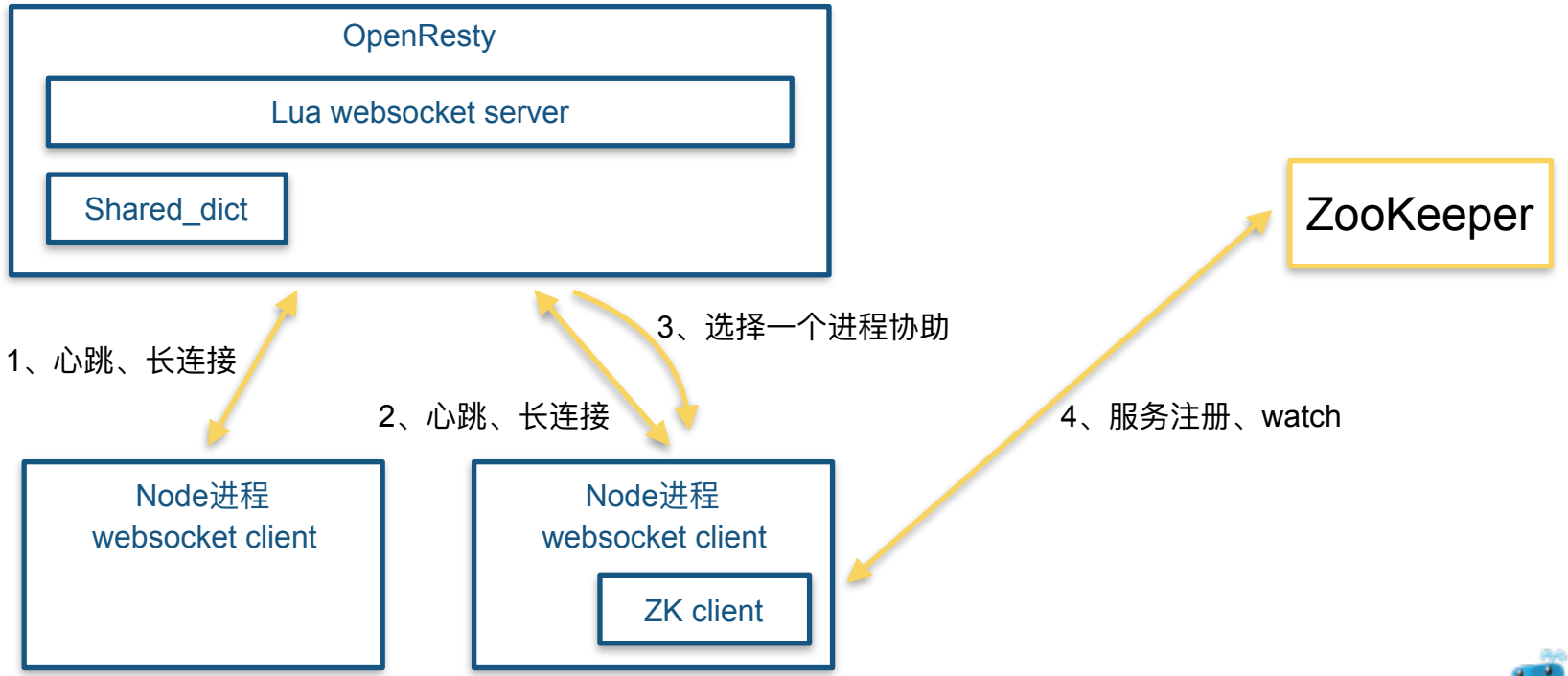

在线运行平台

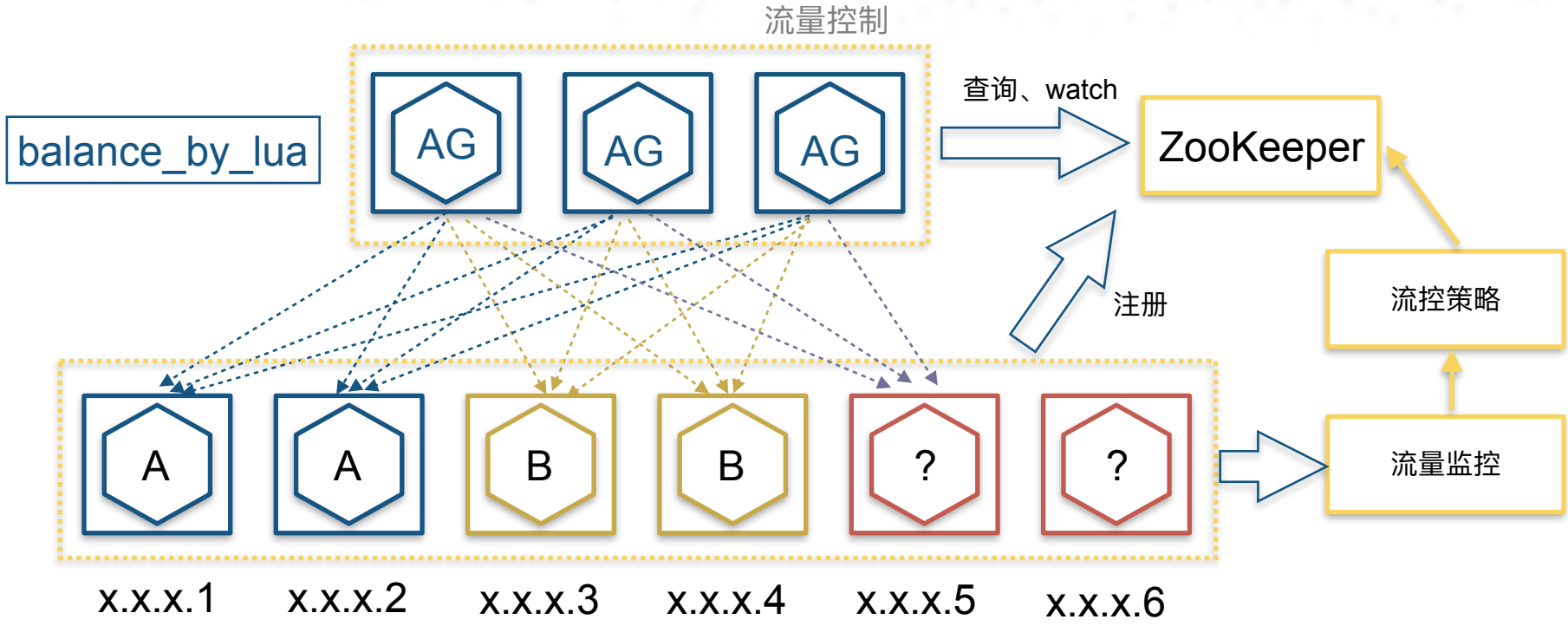
Build&
Deploy




Git代码

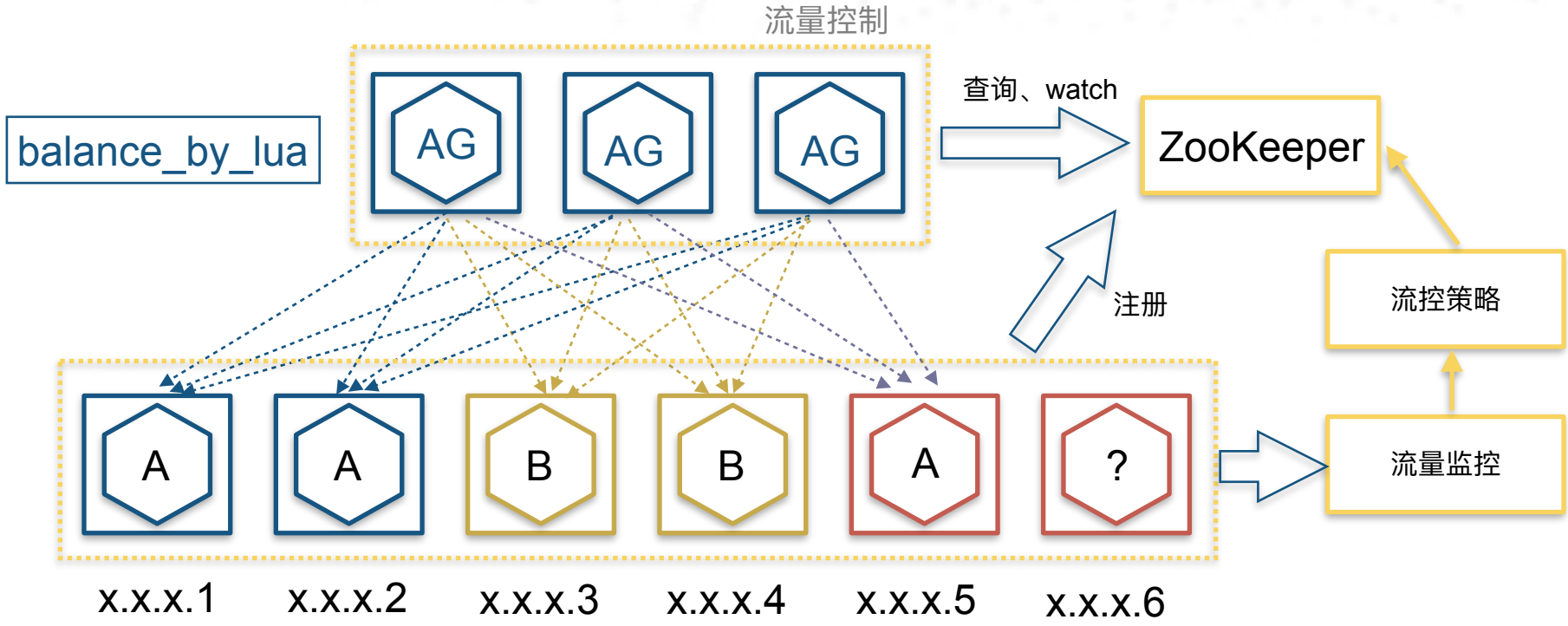
## 服务注册和发现


OpenResty
Lua websocket server
Shared_dict
ZooKeeper
1、心跳、长连接
2、心跳、长连接
3、选择一个进程协助
4、服务注册、watch
Node进程
websocket client
Node进程
websocket client
ZK client




扩容前：ServA:{list:[x.x.x.1,x.x.x.2]}

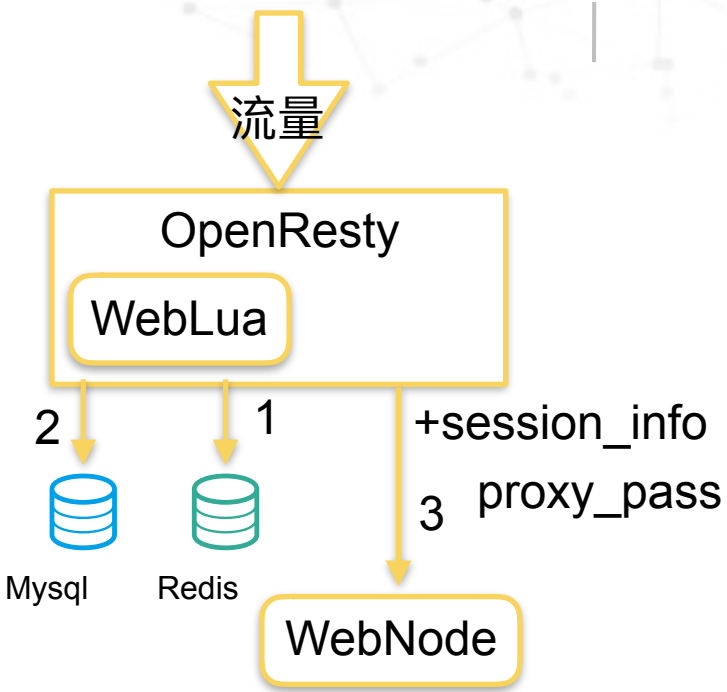
扩容后：ServA:{list:[x.x.x.1,x.x.x.2],auto:[x.x.x.5]}



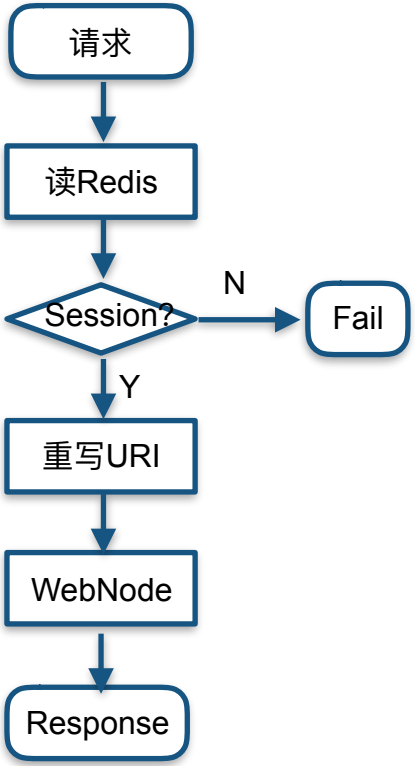

扩容前：ServA:{list:[x.x.x.1,x.x.x.2]}

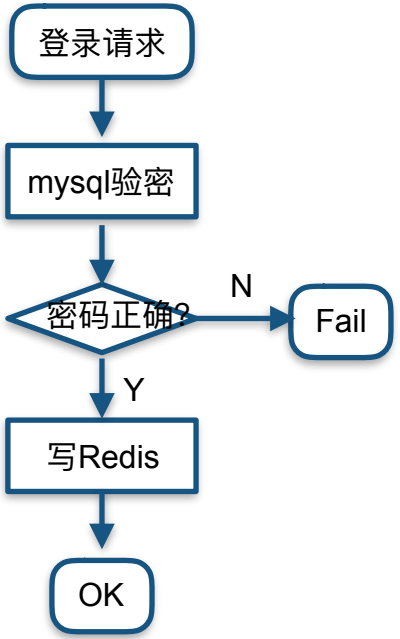
扩容后：ServA:{list:[x.x.x.1,x.x.x.2],auto:[x.x.x.5]}

## 统一鉴权

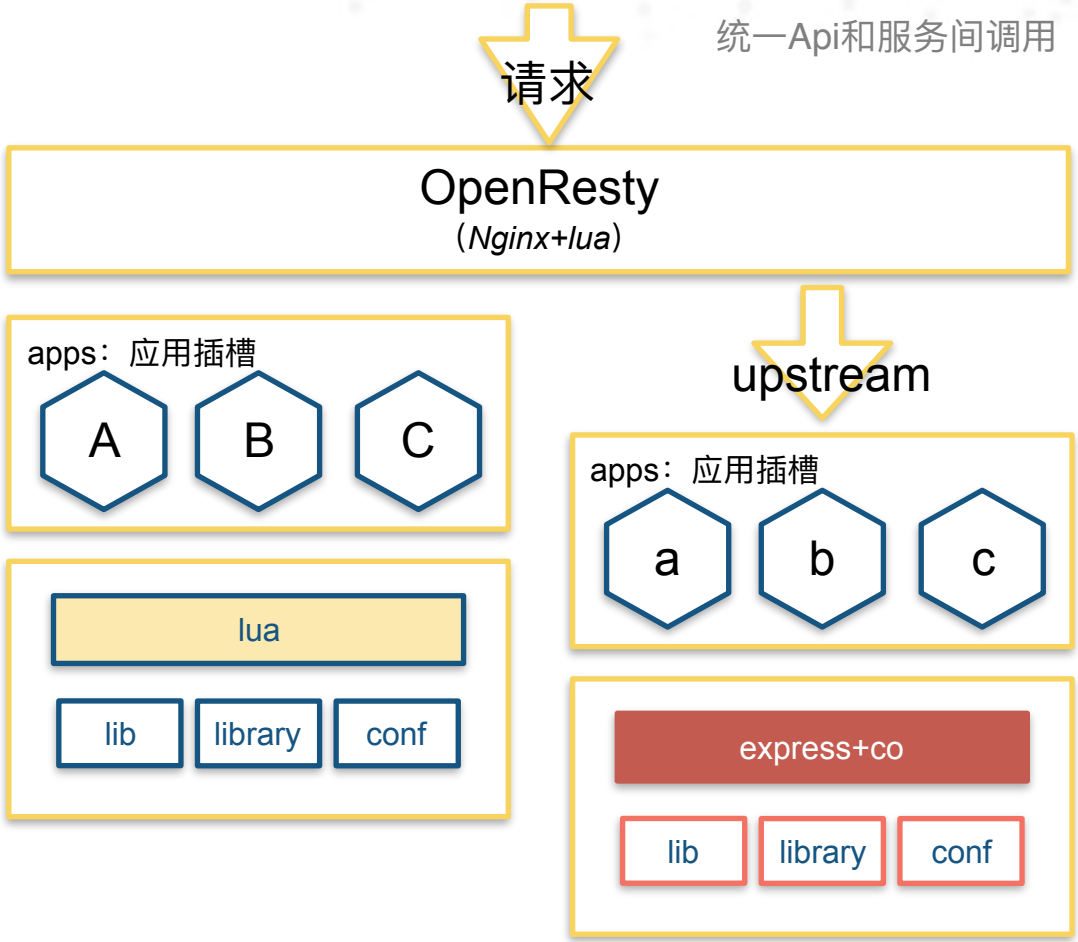



1、session存取
2、密码验证
3、请求执行



统一Api和服务间调用




| | 场景 | 统一api方式 |
|---|---|---|
| 服务单元内 | lua app间调用 | lua require |
| | Node app间调用 | node require |
| | lua和node调用 | http:127.0.0.1:port |
| 跨服务单元 | 内网 | 服务发现，内网http请求 |
| | 公网 | 域名Http服务 |

app::action::fname(params)

个推 GeTui.com

部署实践

Web

OpenResty

upstream

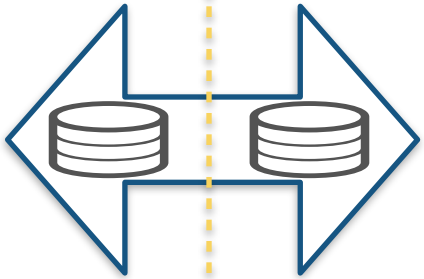
Node apps

B C

express+co

lib library

网闸



解决方案：

只要增加一种通过DB进行中转的API调用方式。

OpenResty

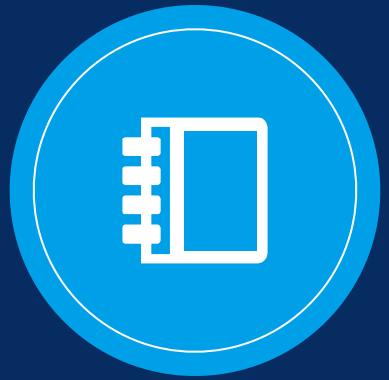
upstream

Node apps

a 依赖B、C

express+co

lib library



# Q&A

# 欢迎互撩


个推 GeTui.com