基于Spring Cloud的Apollo配置中心介绍

宋顺
携程框架研发部

Overview

- What is Apollo
- Why Apollo
- Apollo at a glance
- Apollo in depth
- Future

一个小故事

- 小A是XX团队主力开发，有一天产品说要上线一个迪士尼门票内购功能
- 由于迪士尼门票很火爆，产品一拍脑袋说，每个用户限购5张！
- 于是小A在代码里是这么写的

```
private static final int MAX_QTY_PER_USER = 5;//产品需求限购5张
if (qty > MAX_QTY_PER_USER) {
  throw new IllegalStateException(
          String.format("每个用户最多购买%d张!", MAX_QTY_PER_USER));
}
```

一个小故事

- 第二天中午，由于内购实在太火爆，产品急匆匆的跑过来对小A说，赶紧改成每人1张！
- 小A只好放弃了午饭，改代码、回归测试、上线，整整花了1个小时才搞定。。。

一个小故事

- 小B是YY团队主力开发，有一天产品说要上线一个欢乐谷门票内购功能
- 由于欢乐谷门票很火爆，产品一拍脑袋说，每个用户限购5张！
- 小B吸取了小A的教训，二话不说把配置写在了Apollo配置中心

Key	Value	备注
max-qty-per-user	5	产品需求限购5张

一个小故事

- 第二天中午，由于内购实在太火爆，产品急匆匆的跑过来对小B说，赶紧改成每人1张！
- 小B不紧不慢的说：10秒内搞定~

What is Apollo

- 携程框架部门推出的应用配置中心
- 支持4个维度管理配置（Key-Value）
    - application (应用)
    - environment (环境)
    - cluster (集群)
    - namespace (命名空间)

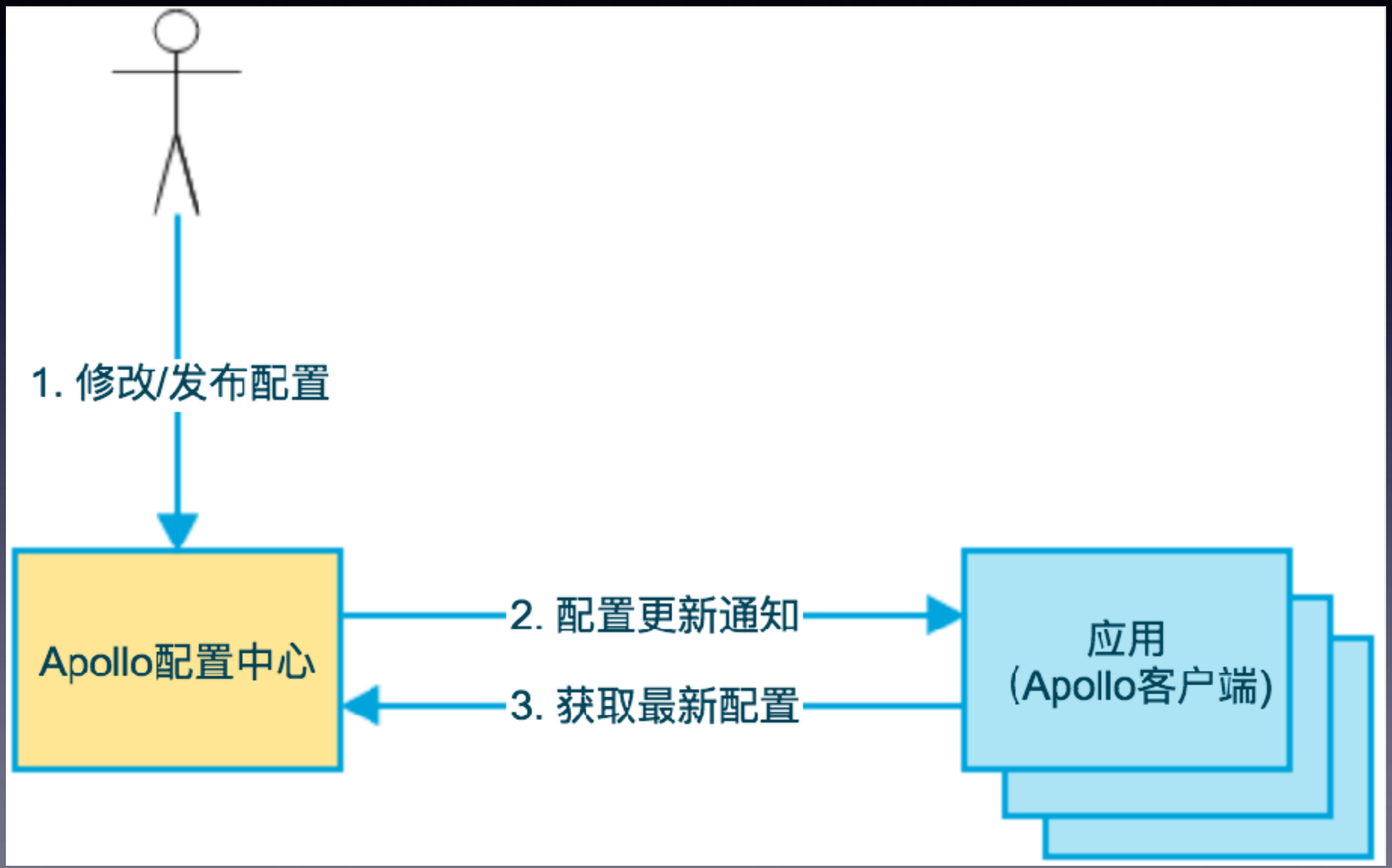
What is Configuration

- 配置是独立于程序的只读变量
  - DB Connection Str、Thread Pool Size、Buffer Size、Request Timeout、Feature Switch、Server Urls等
- 配置伴随应用的整个生命周期
  - 启动时读取配置，运行时根据配置调整行为
- 配置可以有多种加载方式
  - 程序内部hard code，配置文件，环境变量，启动参数，基于数据库等
- 配置需要治理
  - 权限控制、发布审核
  - 不同环境、集群配置管理
  - 公共组件配置管理

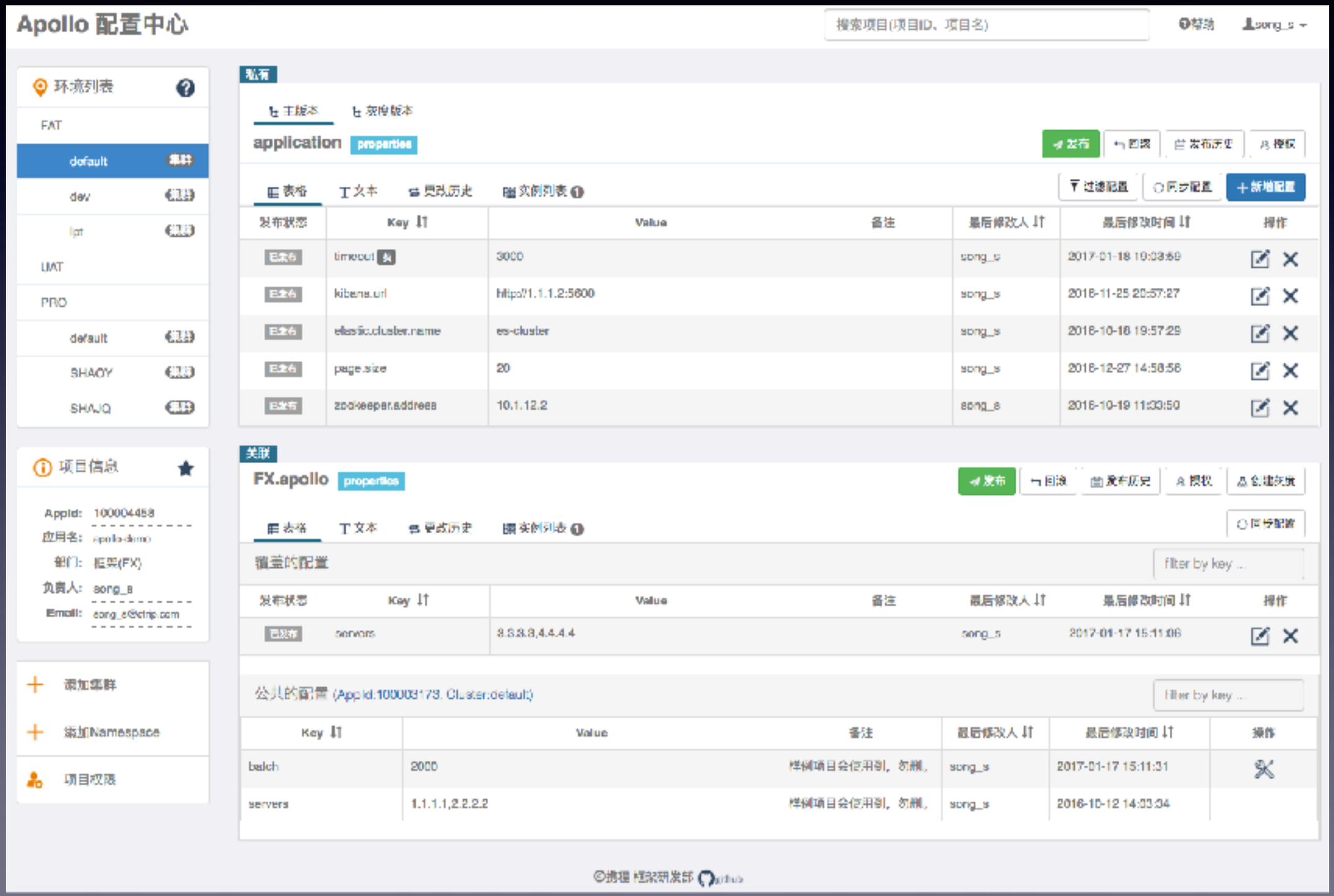
Why Apollo

- 有治理能力的配置管理平台
  - 统一管理不同环境、不同集群的配置
  - 配置修改实时生效（热发布）
  - 版本发布管理
  - 灰度发布
  - 权限管理、发布审核、操作审计
  - 客户端配置信息监控
  - Java，.Net原生客户端，Spring支持

Apollo at a glance



Apollo at a glance



Apollo at a glance

- 添加/修改配置项

Apollo 配置中心

应用ID/应用名 Go 帮助 song_s

环境列表

FAT

UAT

PRO

default 集群

SHAOY 集群

SHAJQ 集群

项目信息

AppId: 100004458

应用名: apollo-demo

部门: 框架(FX)

负责人: song_s

负责人Email: song_s@ctrip.com

application properties

发布 回滚 发布历史 授权 同步配置

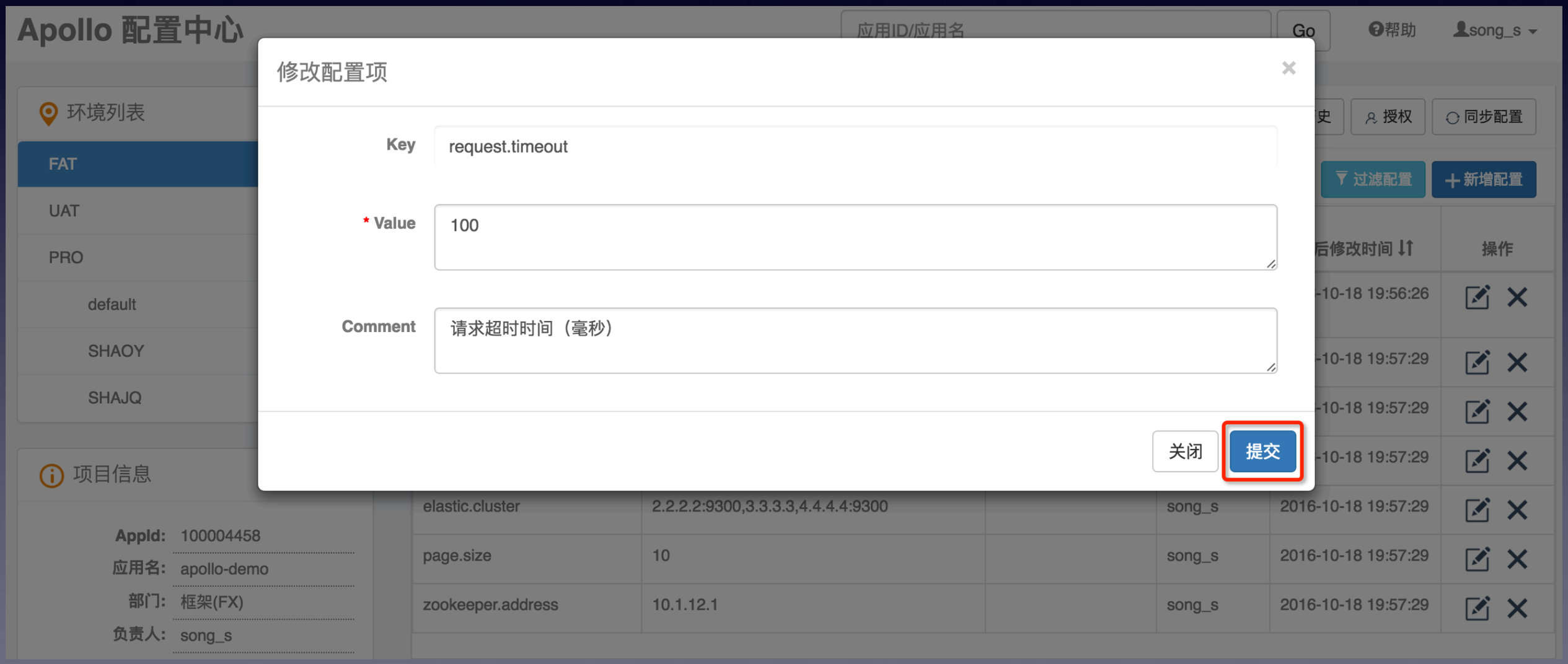
表格 文本 更改历史 实例列表 4

过滤配置 新增配置

Key	Value	备注	最后修改人	最后修改时间	操作
request.timeout	200	请求超时时间（毫秒）	song_s	2016-10-18 19:56:26	
kibana.url	http://1.1.1.2:5601		song_s	2016-10-18 19:57:29	
elastic.document.type	biz		song_s	2016-10-18 19:57:29	
elastic.cluster.name	es-cluster		song_s	2016-10-18 19:57:29	
elastic.cluster	2.2.2.2:9300,3.3.3.3,4.4.4.4:9300		song_s	2016-10-18 19:57:29	
page.size	10		song_s	2016-10-18 19:57:29	
zookeeper.address	10.1.12.1		song_s	2016-10-18 19:57:29	

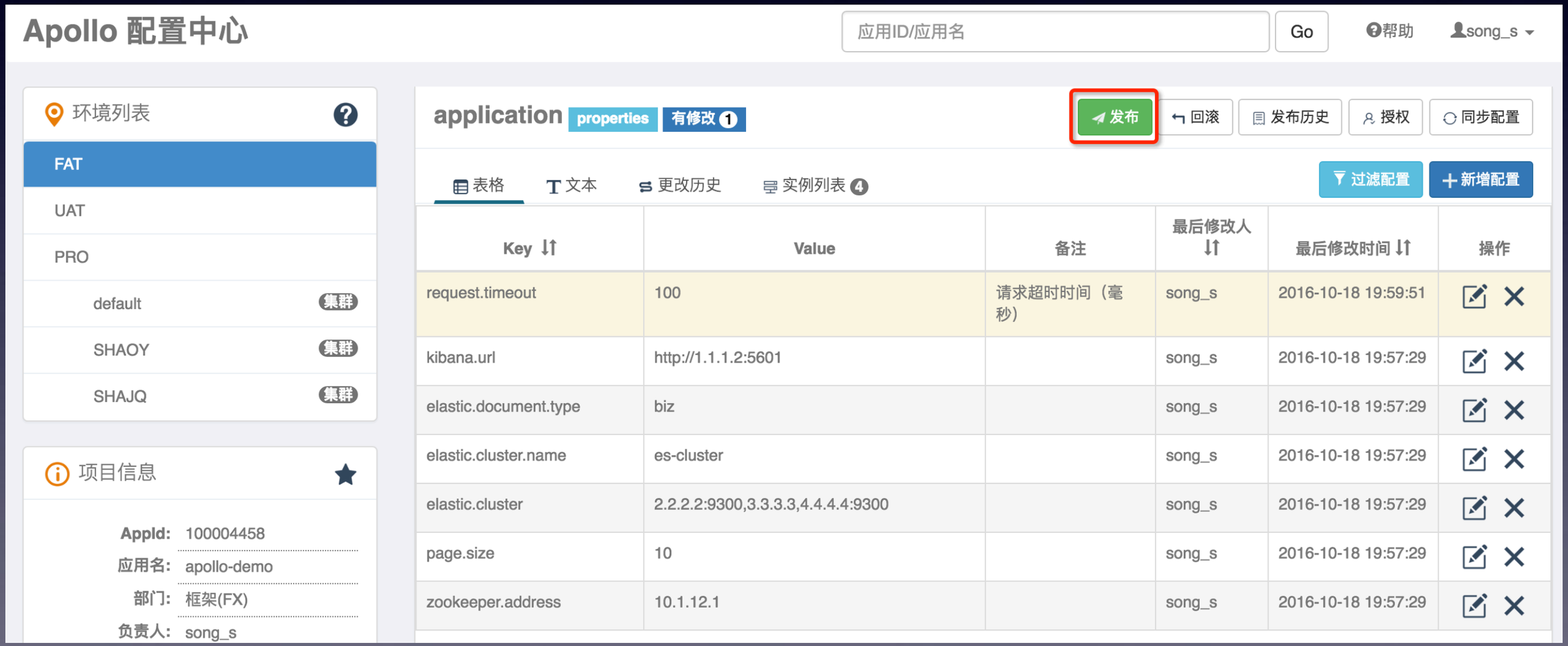
Apollo at a glance

- 添加/修改配置项



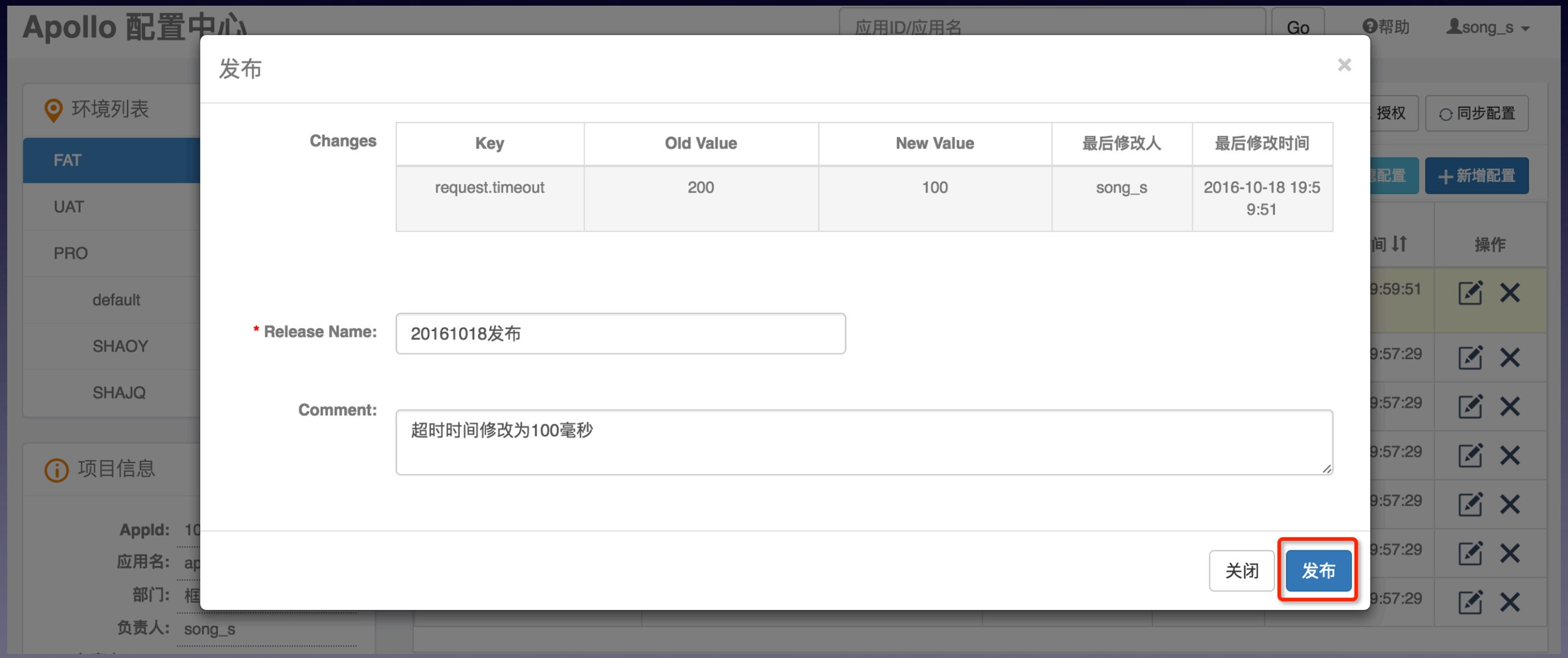
Apollo at a glance

- 发布配置



Apollo at a glance

- 发布配置



Apollo at a glance

- 客户端获取配置（Java API样例）

```
Config config = ConfigService.getAppConfig();
Integer defaultRequestTimeout = 200;
Integer requestTimeout =
        config.getIntProperty("request.timeout", defaultRequestTimeout);
```

Apollo at a glance

- 客户端监听配置变化（Java API样例）

```
Config config = ConfigService.getAppConfig();
config.addChangeListener(new ConfigChangeListener() {
  @Override
  public void onChange(ConfigChangeEvent changeEvent) {
    for (String key : changeEvent.changedKeys()) {
      ConfigChange change = changeEvent.getChange(key);
      System.out.println(String.format(
        "Found change - key: %s, oldValue: %s, newValue: %s, changeType: %s",
        change.getPropertyName(), change.getOldValue(),
        change.getNewValue(), change.getChangeType()));
    }
  }
});
```

Apollo at a glance

- Spring集成样例

```
@Configuration
@EnableApolloConfig
public class AppConfig {}
```

```
@Component
public class SomeBean {
  @Value("${request.timeout:200}")
  private int timeout;

  @ApolloConfigChangeListener
  private void someChangeHandler(ConfigChangeEvent changeEvent) {
    if (changeEvent.isChanged("request.timeout")) {
      refreshTimeout();
    }
  }
}
```

Apollo in depth

- Core Concepts
  - application (应用)
    - 使用配置的应用
    - 有唯一标识appId：
      - Java: classpath:/META-INF/app.properties -> app.id
      - .Net: app.config -> AppID
  - environment (环境)
    - 配置对应的环境
    - DEV, FAT, UAT, PRO：
      - server.properties -> env
      - C:\opt\settings\server.properties或/opt/settings/server.properties

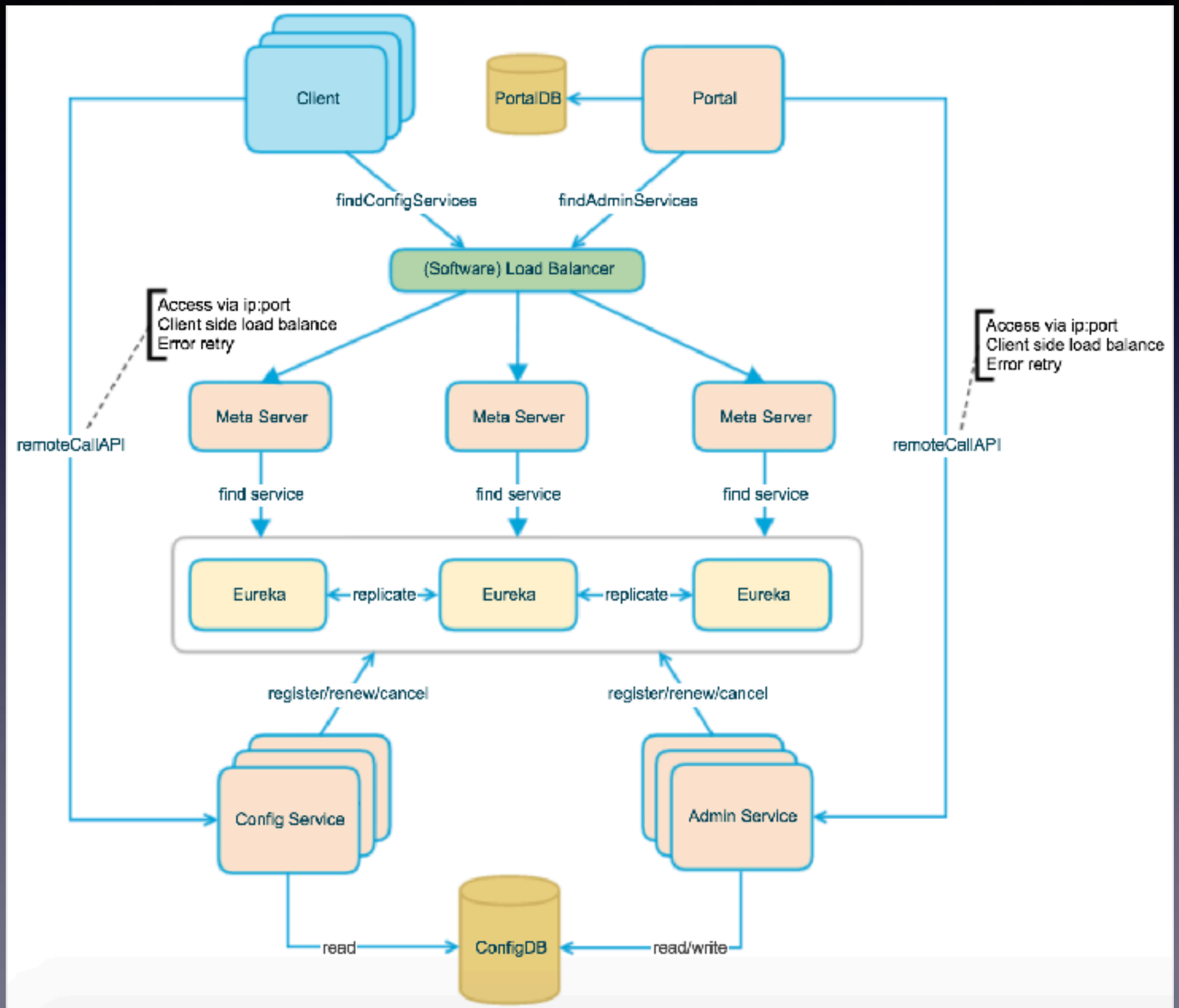
Apollo in depth

- Core Concepts - cluster (集群)
  - 一个应用下不同实例的分组
  - 对不同的cluster，可以有不一样的配置
    - 比如zk地址针对上海机房和成都机房可以有不一样的配置
- 默认数据中心作为cluster
  - server.properties -> idc
  - C:\opt\settings\server.properties或/opt/settings/server.properties

Apollo in depth

- Core Concepts - namespace (命名空间)
  - 一个应用下不同配置的分组
  - 应用默认有自己的配置namespace - application
  - 也可以使用公共组件的配置namespace
    - 如RPC，DAL等
    - 可以通过继承方式对公共组件的配置做调整，如DAL的初始数据库连接数

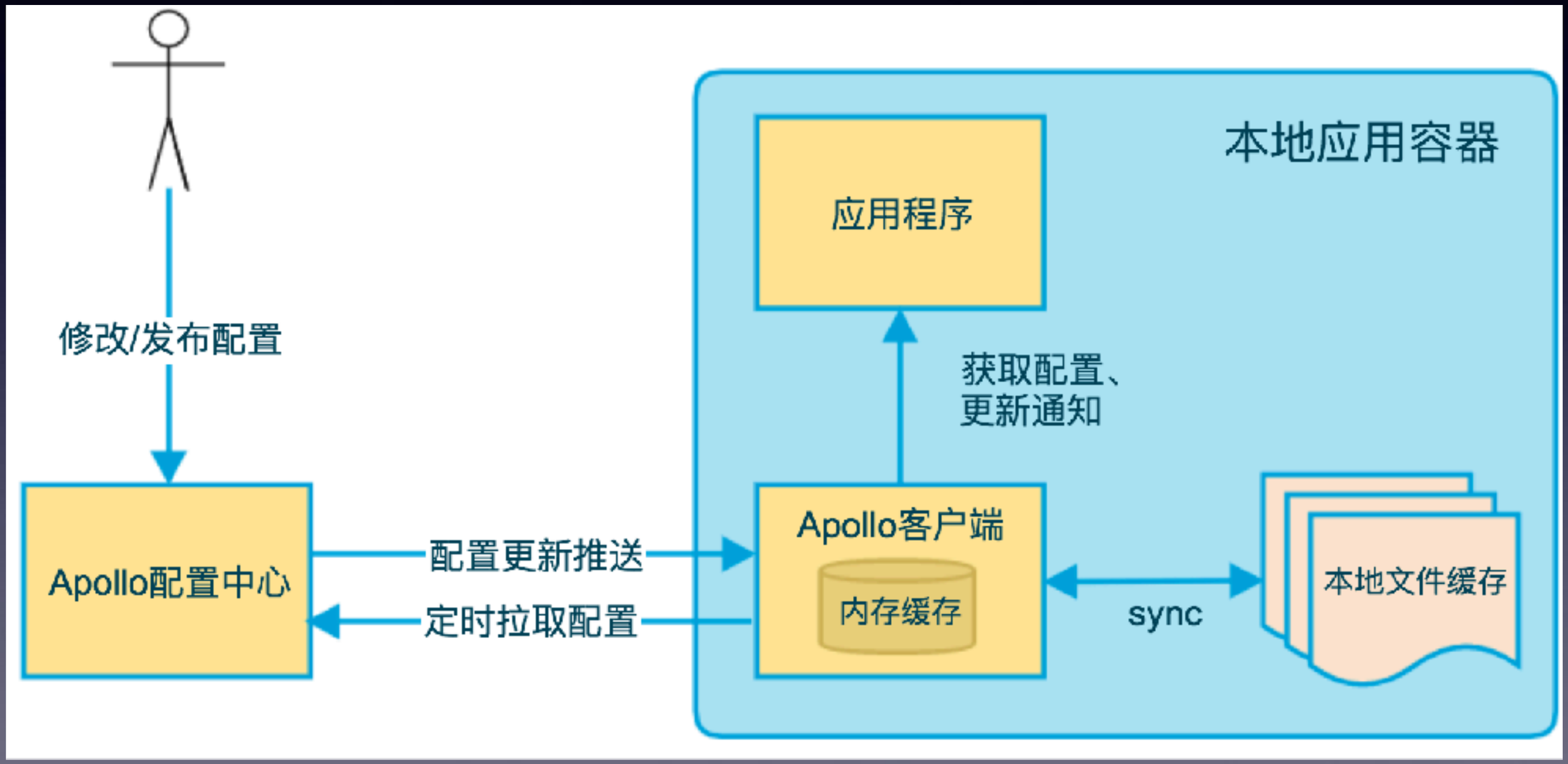
总体设计


Client
PortalDB
Portal
findConfigServices
findAdminServices
(Software) Load Balancer
Access via ip:port
Client side load balance
Error retry
Meta Server
Meta Server
Meta Server
remoteCallAPI
find service
find service
find service
Eureka
replicate
Eureka
replicate
Eureka
register/renew/cancel
register/renew/cancel
Config Service
Admin Service
read
ConfigDB
read/write
Access via ip:port
Client side load balance
Error retry
remoteCallAPI

Why Eureka?

- 完整的 Service Registry 和 Service Discovery 实现
- 和 Spring Cloud 无缝集成
- Open Source

客户端设计


修改/发布配置
Apollo配置中心
配置更新推送
定时拉取配置
本地应用容器
应用程序
获取配置、
更新通知
Apollo客户端
内存缓存
sync
本地文件缓存

配置更新推送

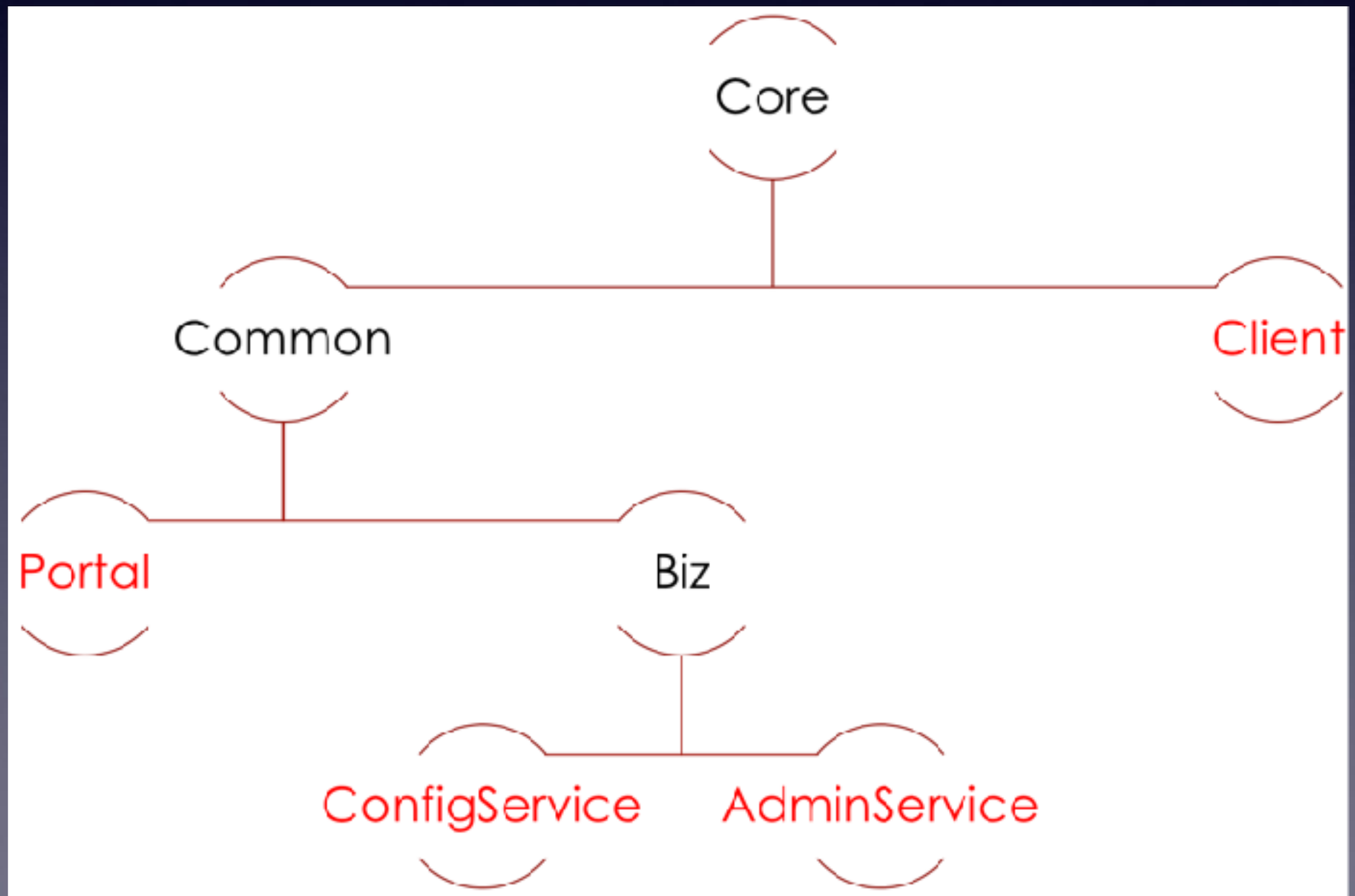
- 客户端Http Long Polling
  - 服务端保持30秒
  - 客户端断开自动重连
- 服务端async servlet
  - Spring DeferredResult

可用性考虑

场景	影响	降级	原因
某台config service下线	无影响		Config service无状态，客户端重连其它config service
所有config service下线	客户端无法读取最新配置，Portal无影响	客户端重启时,可以读取本地缓存配置文件	
某台admin service下线	无影响		Admin service无状态，Portal重连其它admin service
所有admin service下线	客户端无影响，portal无法更新配置		
某台portal下线	无影响		Portal域名通过slb绑定多台服务器，重试后指向可用的服务器
全部portal下线	客户端无影响，portal无法更新配置		
某个数据中心下线	无影响		多数据中心部署，数据完全同步，Meta Server/Portal域名通过slb自动切换到其它存活的数据中心

Contribute to Apollo

- https://github.com/ctripcorp/apollo
- 服务端基于Spring Cloud和Spring Boot开发



Future

- 配置支持简单格式，如数字、日期等
- 配置支持复杂格式，如Table、Tree等

Summary

- What is Apollo
- Why Apollo
- Apollo at a glance
- Apollo in depth
- Future

Q&A