融汇贯通-支付系统大数据风控实战课程

==楼兰==

一、课程介绍

1、课程结构介绍

项目： 基于大数据技术，对一个复杂的支付系统提供统一、全面、高效的风险控制服务。

为什么选这个项目？1、很轻松的切入业务场景。2、非常具有代表性。

课程特点： 1、真实。2、精简。3、细致。4、实用。

课程收获：1、大数据相关组件如何整合工作。2、从掌握工具技术，转变到真正解决实际问题。

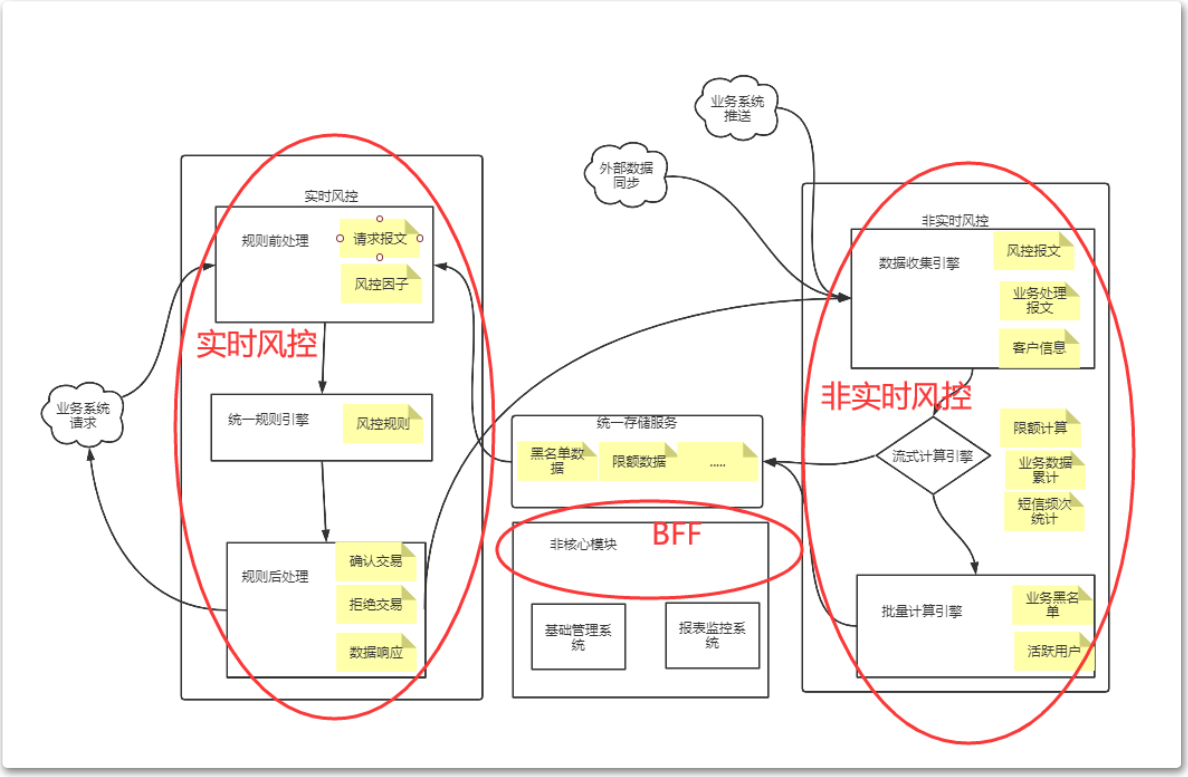
适合人群: 1、有J2EE的开发经验。2、对相关大数据组件比较熟悉。3、希望掌握大数据项目落地实践

课程体系：大数据架构技术课程体系，DDD领域驱动设计。

2、项目背景介绍

项目： 对一个复杂的支付系统提供统一、全面、高效的风险控制服务。

特点：1、数据全，2、响应快。



事件风暴会议

五种典型规则来进行实现。

1、简单型规则： 直接读取请求报文进行规则判断。

规则示例：AQ001：对于交易渠道为DPAY的所有交易，如果交易手机号和银行签约手机号相同，则不做任何规则限制。

2、数据画像型规则： 需要在请求报文基础上添加一些风控因子进行补充判断。

规则示例：LG001：外部导入一批黑名单数据，黑名单用户禁止登录。

3、累计型数据规则： 需要对用户以往的交易行为进行累计计算的规则。

规则示例：LJ001：设定用户的日交易限额为500，超过日交易额的用户，禁止当日的所有交易。

4、批量计算型规则：需要对用户以往的交易行为进行批量统计的规则。

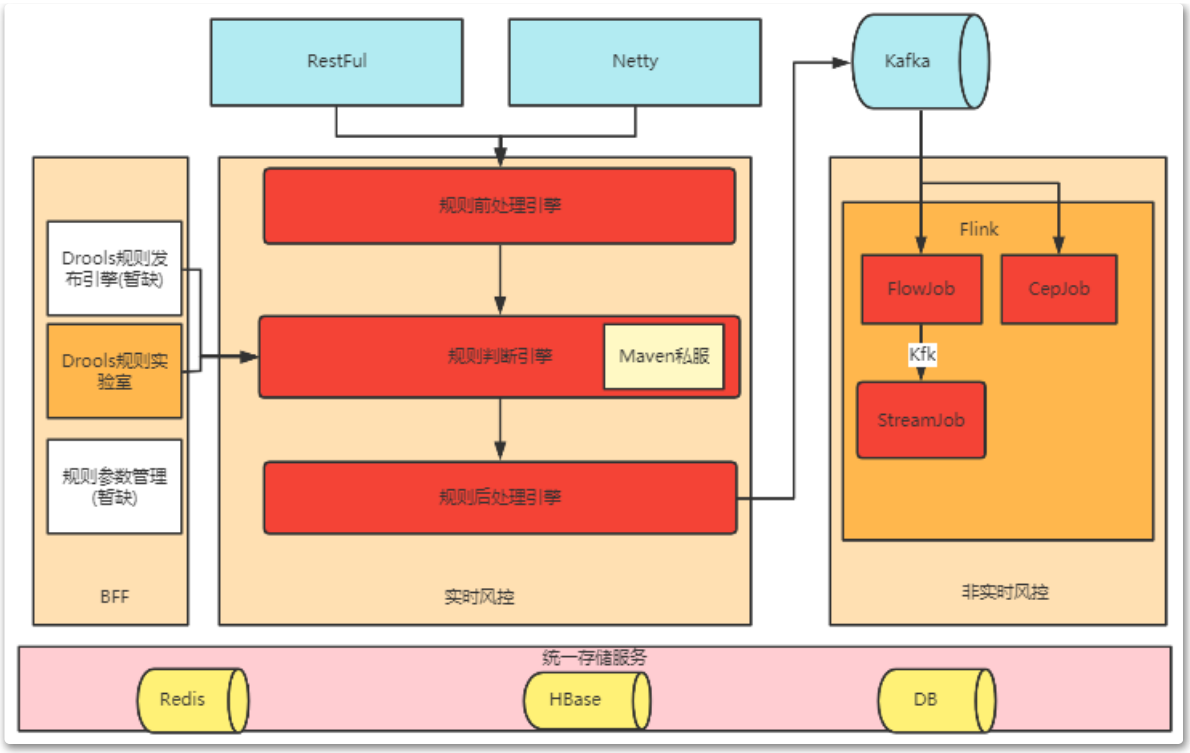
规则示例：LJ002：同一个手机号，在三天内支付次数超过10次，支付总金额不超过100元，禁止支付交易12小时。

5、复杂事件型规则： 需要对用户以往行为进行组合甄别的规则。

规则示例：LG002：同一个手机号，在一天内，连续登陆失败5次，则锁定账户，3天内禁止登录。

3、技术组件介绍

需要重点考虑的几个因素： 数据量大小，业务需求，技术成熟度，维护成本。



项目环境： Kafka + HBase + Redis + Flink + JDBC(可选) + Nexus(Maven私服)

实时风控： SpringBoot + Netty + Drools

非实时风控： Flink

二、项目演示

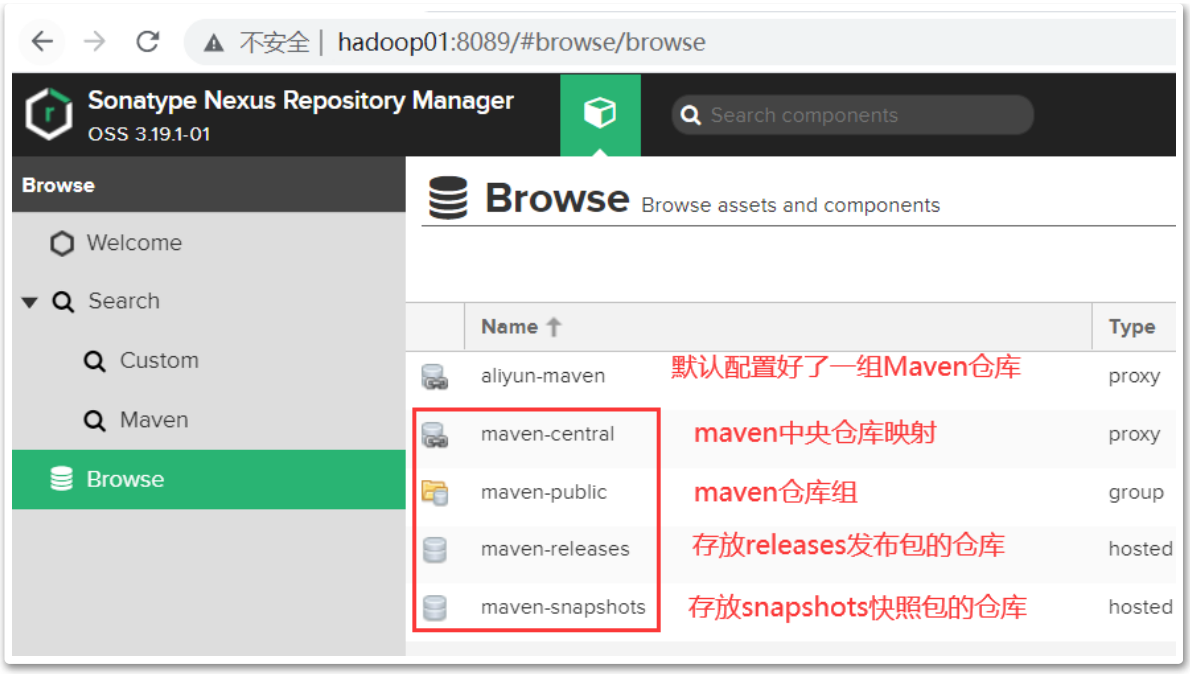
1、搭建Maven私服

安装步骤：

1、下载nexus安装包。 课程中提供了nexus-3.19.1-01-unix.tar.gz用于Unix环境搭建。

2、解压后，获得两个目录 nexus-3.19.1-01 和 sonatype-work 。 直接执行 nexus-3.19.1-01/bin/nexus run即可启动nexus服务。 启动完成后，即可访问目标主机的8081端口。访问maven私服。

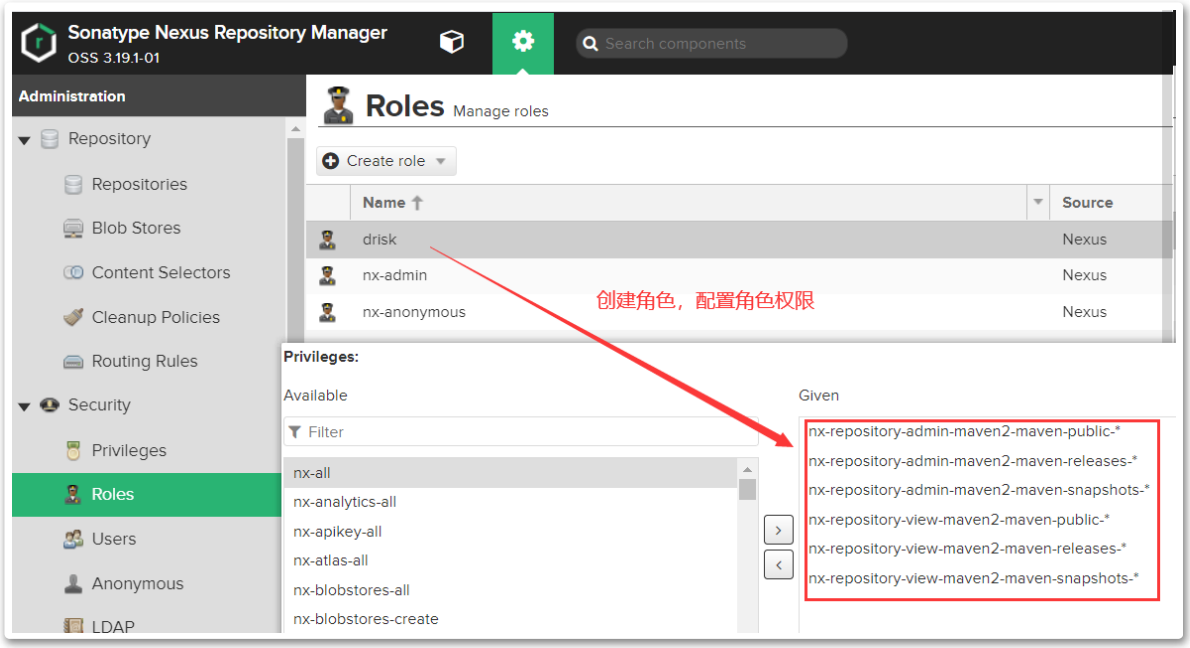
服务默认会启动在8081端口，如果需要调整端口，可以修改nexus-3.19.1-01/etc/nexus-default.properties文件中的端口配置。



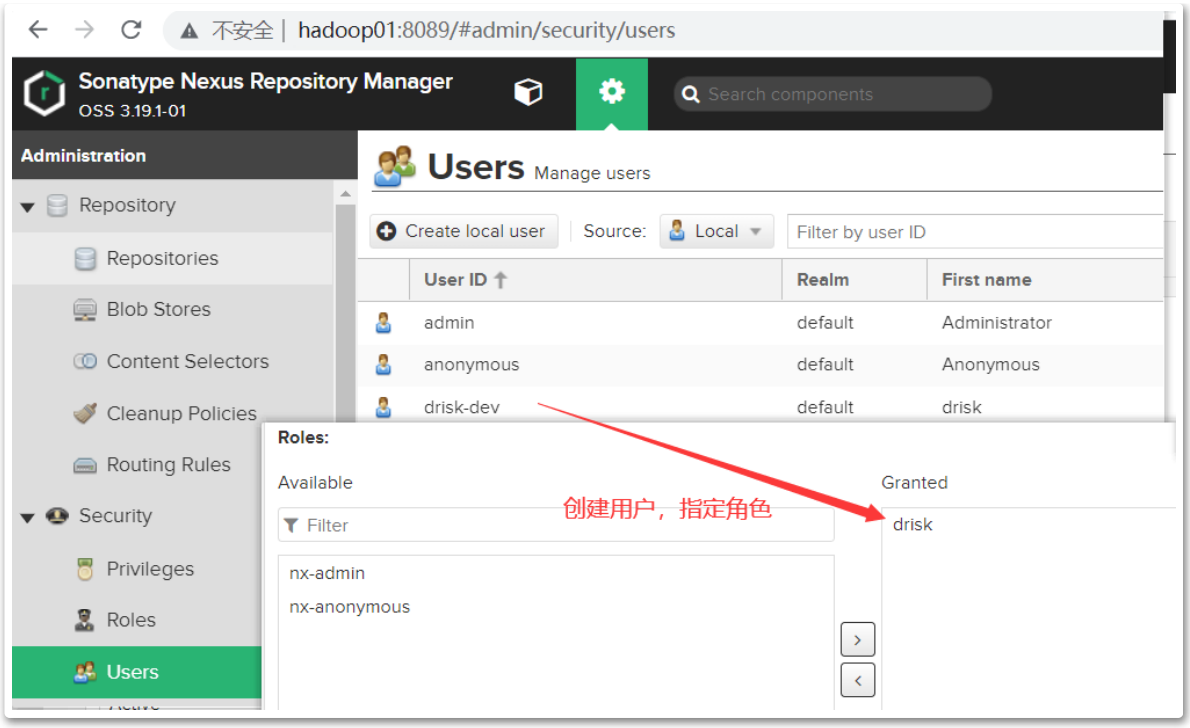
3、配置上传用户

maven中默认有一个admin用户， 密码默认是admin123，可以对仓库进行管理。使用时，通常建议另外配置一个供maven客户端使用的用户来访问jar包上传下载功能。

点击右上角登录按钮，使用admin用户登录。 在设置页面创建drisk用户，分配对maven指定仓库的访问权限。



接下来，创建用户，指定角色。



4、Maven基本配置

Maven服务端完成配置后，就可以在客户端配置使用这个仓库了。配置方式主要是调整maven本地的settings.xml配置文件。主要是配置server和mirrors两个部分。


```
<server>
  <id>drisk-dev</id>
  <username>drisk-dev</username>
  <password>123qweasd</password>
</server>
<server>
  <id>drisk-dev-snapshot</id>
  <username>drisk-dev</username>
  <password>123qweasd</password>
</server>
</servers>

<mirrors>
  <mirror>
    <id>nexus-aliyun</id>
    <mirrorOf>*</mirrorOf>
    <name>Nexus aliyun</name>
    <url>https://maven.aliyun.com/repository/public</url>
  </mirror>
  <mirror>
    <id>drisk-dev</id>
    <mirrorOf>*</mirrorOf>
    <url>http://hadoop01:8089/repository/maven-public/</url>
  </mirror>
    <mirror>
    <id>drisk-dev-snapshot</id>
    <mirrorOf>*</mirrorOf>
    <url>http://hadoop01:8089/repository/maven-public/</url>
  </mirror>
</mirrors>
```

5、接下来，客户端就可以使用mvn指定来上传指定jar包到maven私服上。

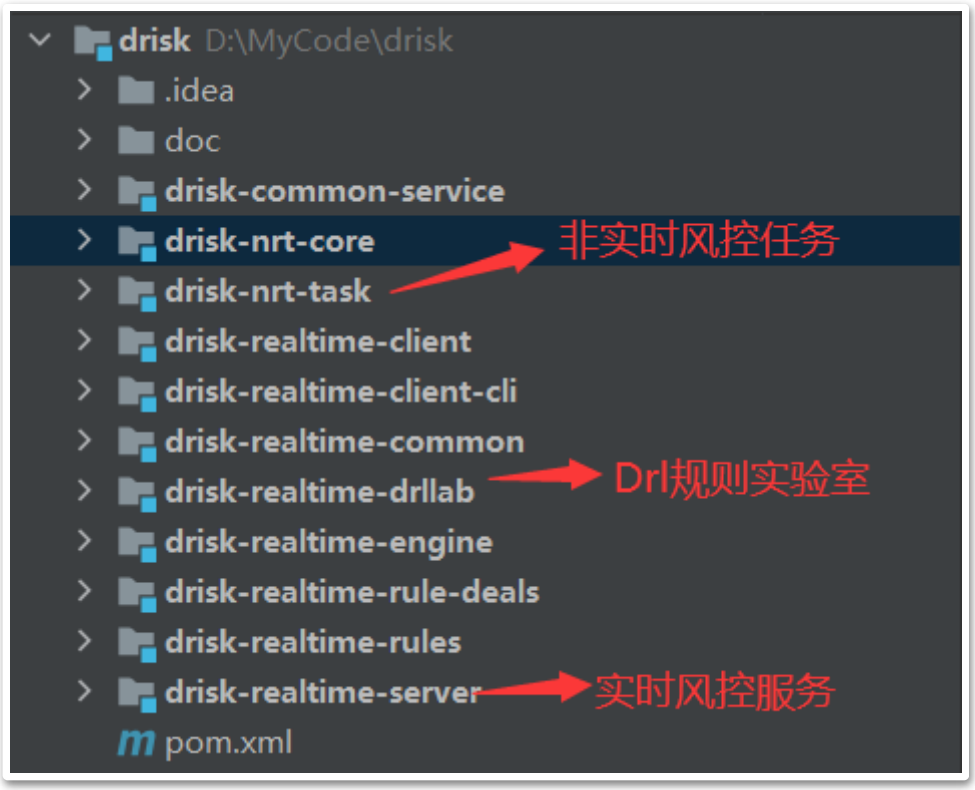
```
E:\a-work\driskInfo>mvn deploy:deploy-file -Dmaven.test.skip=true -
Dfile=./drisk-realtime-rules-1.0.jar -DgroupId=com.roy.drisk -
DartifactId=drisk-rules -Dversion=1.0 -Dpackaging=jar -
Durl=http://hadoop01:8089/repository/maven-releases -DrepositoryId=drisk-dev
```

主要是注意repositoryId属性需要与settings.xml中的mirror的id对应。上传完成后，就可以在maven私服中看到刚上传的jar包。

> 注意：1、未来我们项目的依赖包，以及项目编译后的发布包，都会上传到Maven私服上。私服上没有的jar包，他会自动去maven中央仓库下载。当然，在IDEA中的maven项目，会有不同的配置方式，也不需要手动敲这么长的指令，但是大致是差不多的。

> 2、注意snapshots包和releases包的区别。snapshots包的version以-SNAPSHOT结尾，服务端上会保存每一个版本的快照。不以-SNAPSHOT结尾的包都是releases发布包。服务端上对于发布包，只会保存一个唯一的版本。

2、项目演示

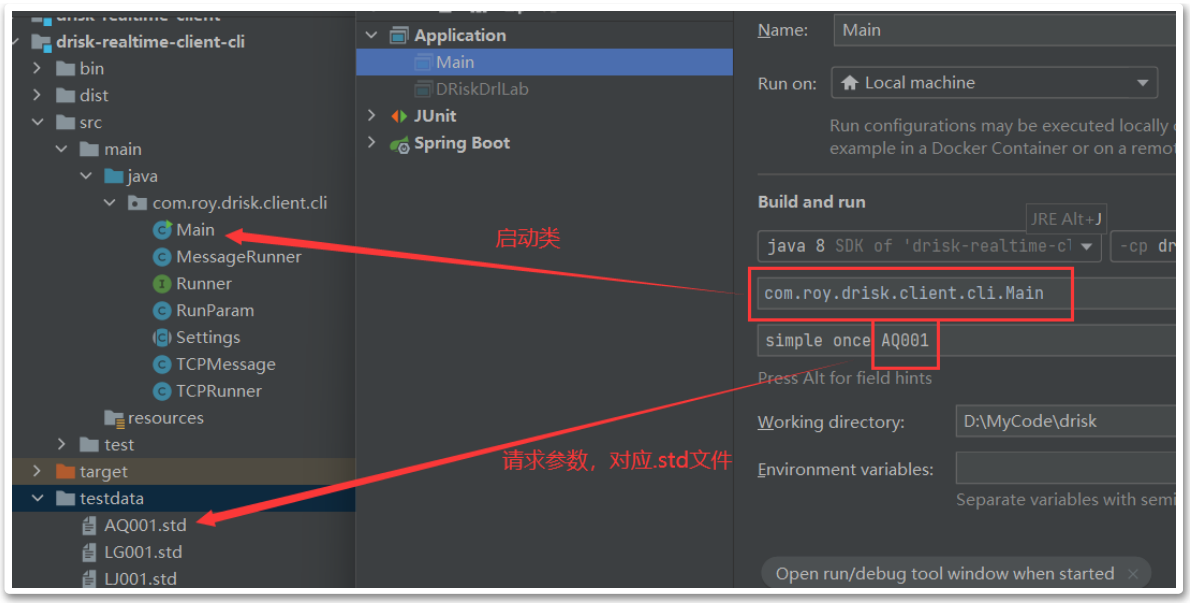


几个重要的配置文件：

connector-${ENV}.properties。配置服务端的数据组件连接地址。其中${ENV}是通过读取操作系统的DRISK_ENV环境变量指定。也可以在应用中通过在System.getProperties中配置 drisk.env 参数指定。如果都没有指定的话，应用会报错的。

请求流程：

客户端的测试包是drisk-realtime-client-cli。通过其中的Main类进行调用。



请求结果

三、重点技术介绍

1、Drools规则引擎介绍

什么是规则引擎？为什么要用规则引擎?

简单来看， 规则引擎就是将一段字符串当作一段逻辑运行。

使用规则引擎，他能够将数据的准备与处理过程分离！！ 解耦。

简单的规则引擎：Aviator 表达式引擎

如何使用Drools规则引擎?

Drools官网： https://www.drools.org/

Drools中文网： http://www.drools.org.cn/

Drools基础使用：

了解Drools的基础使用方法。如何准备数据，如何定义规则文件。

Drools高级使用：

StatefulSession与StatelessfulSession的使用以及区别。

基于Kie组件动态更新Maven仓库中的规则文件。

2、 Flink计算框架

Flink是什么？他能用来干什么？

Flink官网：https://flink.apache.org/zh

Apache Flink 是一个**框架和分布式处理引擎**，用于在无边界和有边界数据流上进行有状态的计算。Flink 能在所有常见集群环境中运行，并能以内存速度和任意规模进行计算。

无界流和有界流： QQ聊天窗口 - 无界流 - 流式计算 - 实时计算 - OLTP

QQ聊天记录 - 有界流 - 批量计算 - 离线计算 - OLAP

Flink是一个流批统一的计算框架。

状态：

Flink运行环境： StandAlone Hadoop-Yarn Mesos

为什么要选择Flink

Hadoop-MapReduce - 离线计算，吞吐量很大，但是计算时间就很长。

Spark： 流批统一的计算框架。

Spark与Flink

对流式数据的处理方式： Spark是以小批量的方式来处理流式数据。

Flink是以流式数据的方式来处理批量量数据。

如何规划Flink的运行资源

对于Flink并行度的规划与管理。

3、其他重点技术介绍

如何选择RPC框架

Netty直连。

心跳、传输协议

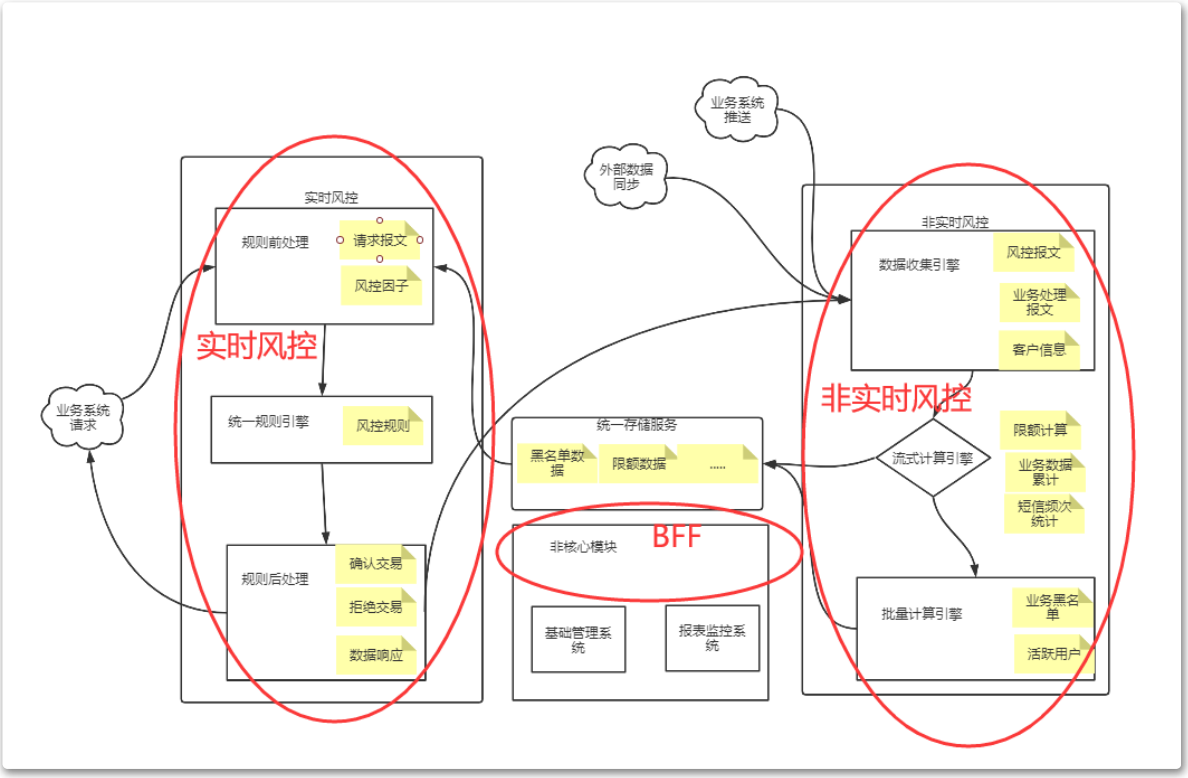
如何选择存储产品

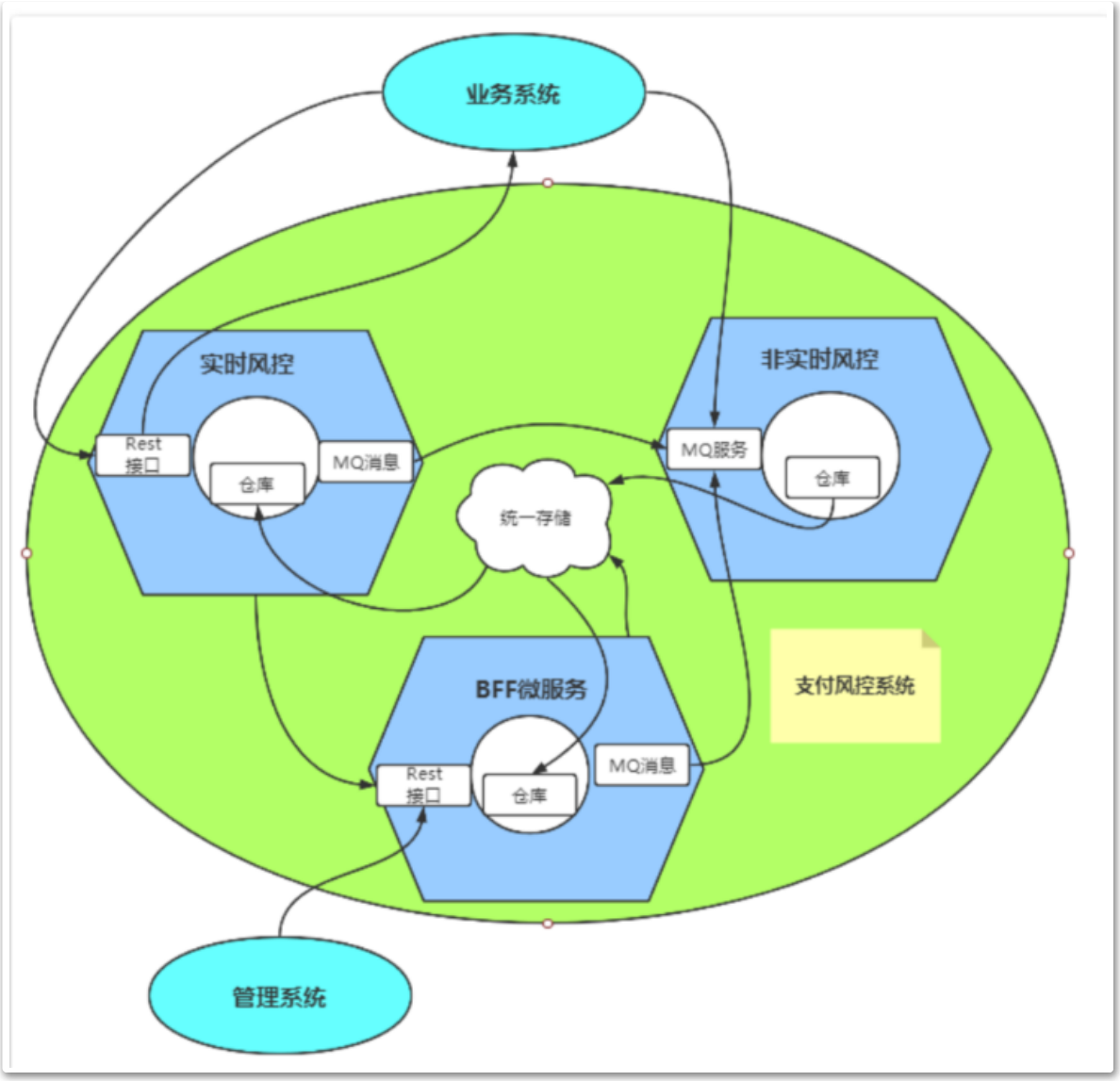
MySQL：数据量比较小，访问也不会太频繁的元数据。

Redis： 数据量比较小，业务价值比较高的数据。

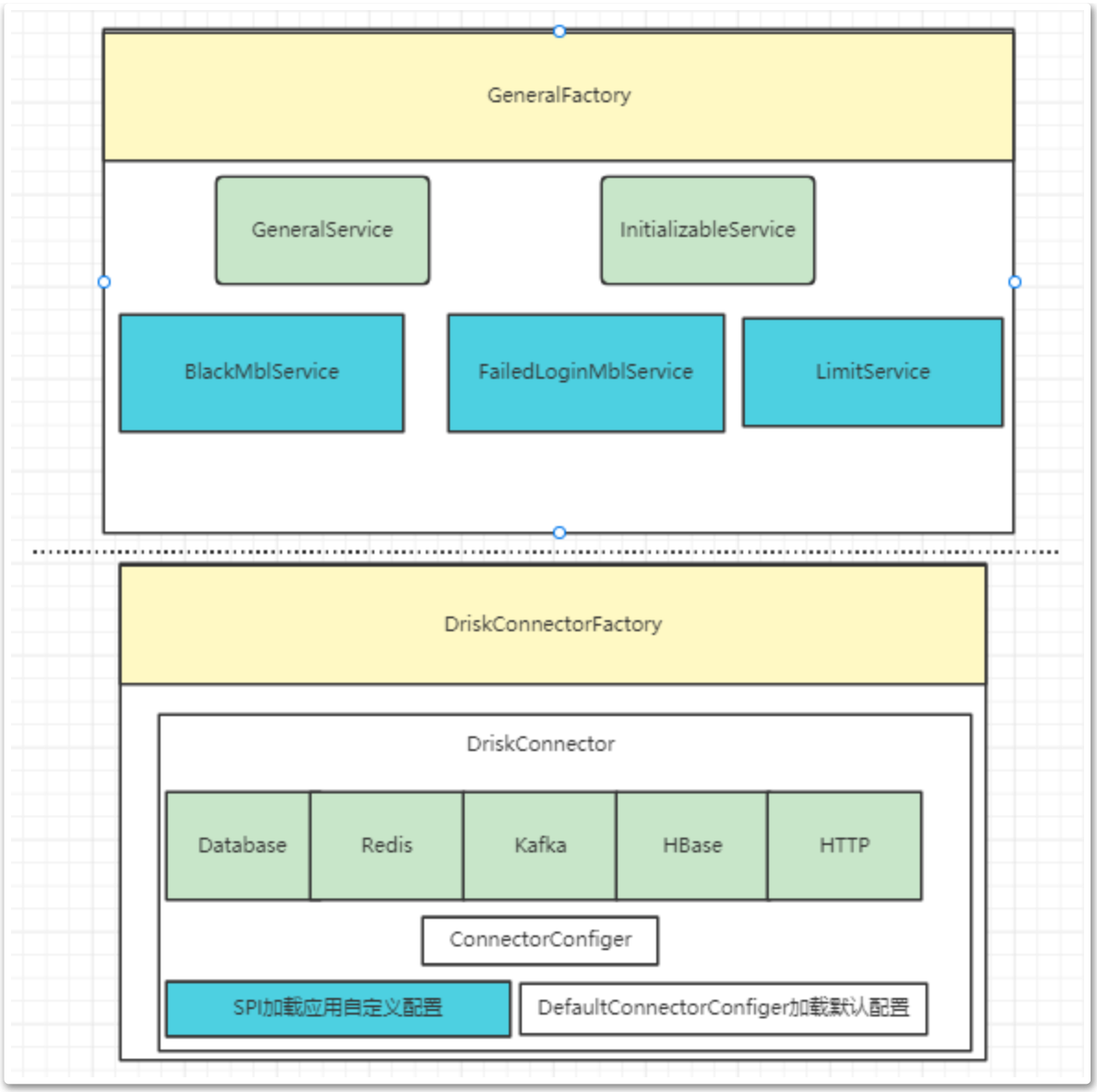
HBase：数据量比较大，业务价值相对比较低的数据。大而全。--clickhouse voltdb es

四、实现流程





1、数据仓库层统一服务化处理



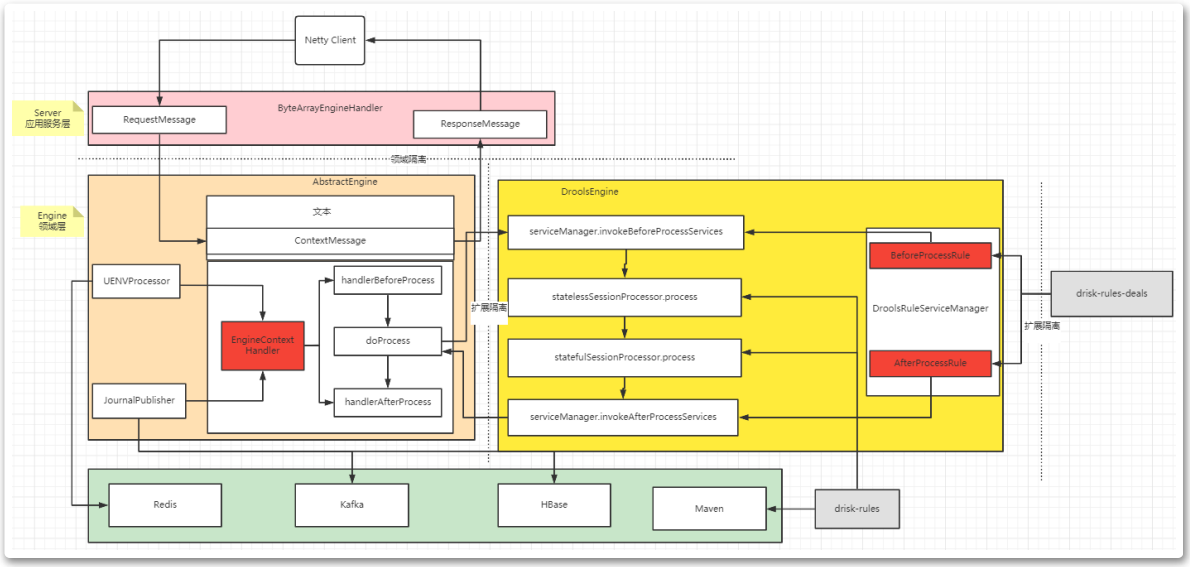
2、实时风控引擎分析

1、如果不想用Netty，如何构建微服务体系？

Server层只处理网络协议，负责对接具体的业务场景，对Engine层形成保护。

2、如果不想用Drools，如何引入新的规则引擎？

3、如何保证规则与数据是同步的？



Netty的负载均衡：

3、简单型及数据画像型规则处理流程分析

1、简单型规则： 直接读取请求报文进行规则判断。

规则示例：AQ001：对于交易渠道为DPAY的所有交易，如果交易手机号和银行签约手机号相同，则不做任何规则限制。

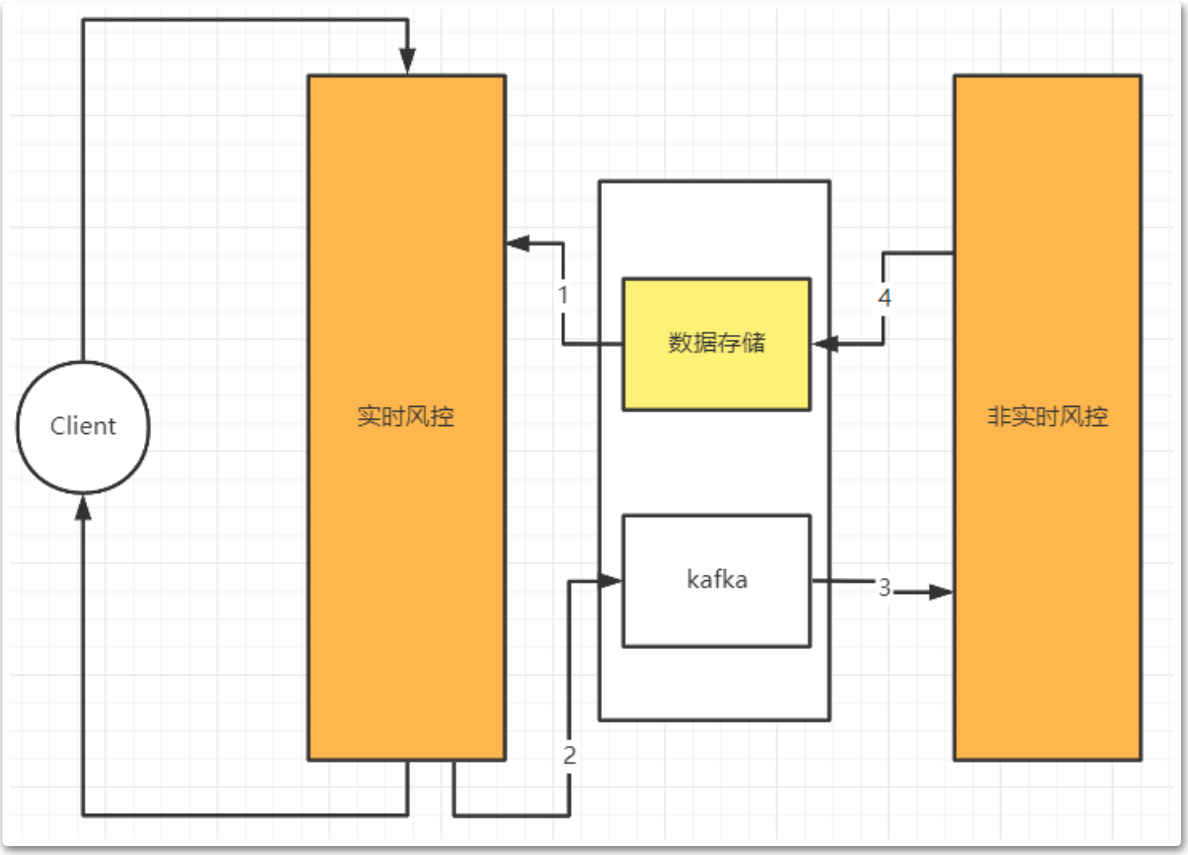
2、数据画像型规则： 需要在请求报文基础上添加一些风控因子进行补充判断。

规则示例：LG001：外部导入一批黑名单数据，黑名单用户禁止登录。

4、累计型规则处理流程及扩展实现

3、累计型数据规则： 需要对用户以往的交易行为进行累计计算的规则。

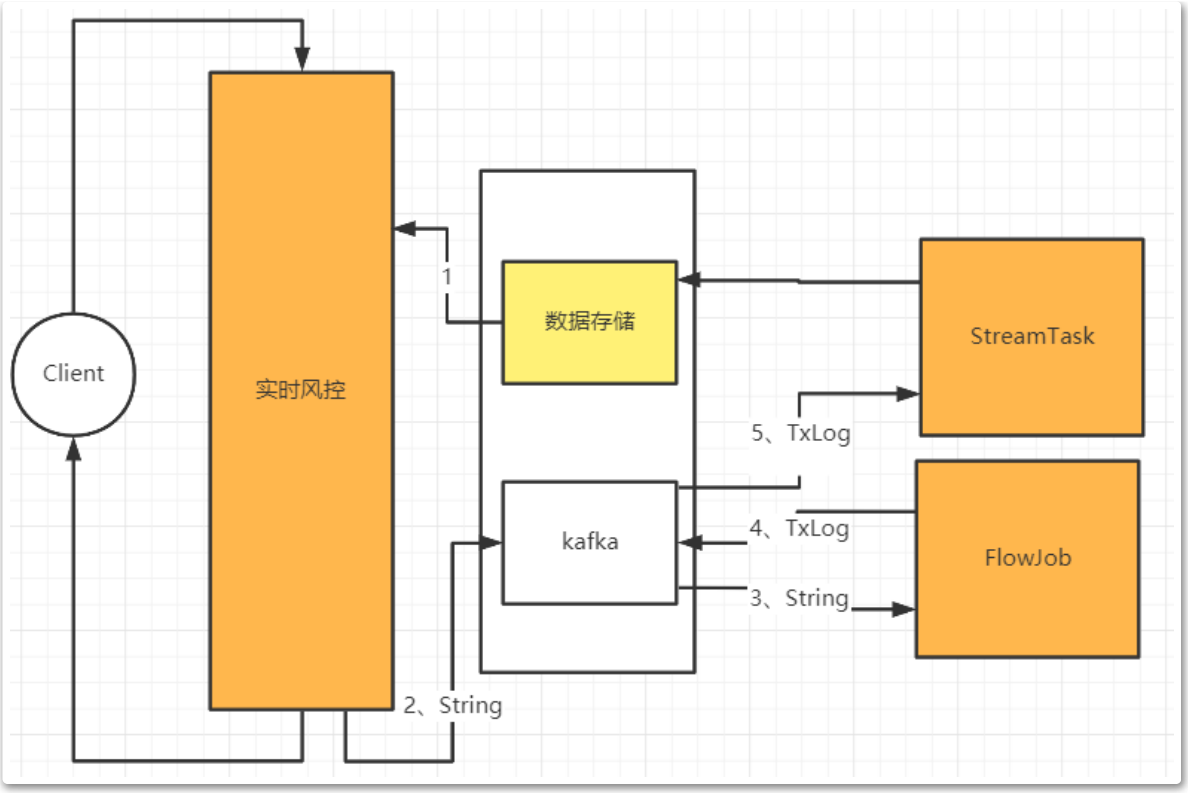
规则示例：LJ001：设定用户的日交易限额为500，超过日交易额的用户，禁止当日的所有交易。



5、批量计算型规则处理流程及扩展实现

4、批量计算型规则：需要对用户以往的交易行为进行批量统计的规则。

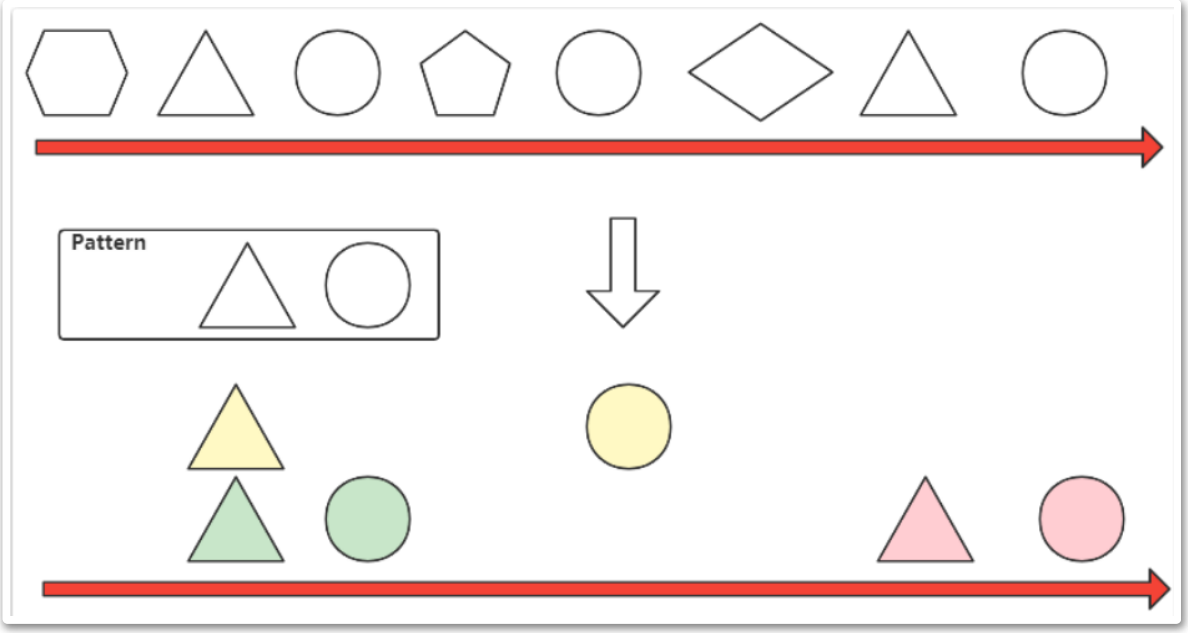
规则示例：LJ002：同一个手机号，在三天内支付次数超过10次，支付总金额不超过100元，禁止支付交易12小时。



6、复杂事件型规则处理流程分析

5、复杂事件型规则： 需要对用户以往行为进行组合甄别的规则。

规则示例：LG002：同一个手机号，在一天内，连续登陆失败5次，则锁定账户，3天内禁止登录。



Pattern的几种匹配类型：

1、量词匹配

2、组合匹配

3、条件匹配 - or　停止条件匹配

4、严格连续匹配

5、松散连续匹配

匹配后的跳过策略：

1、NO_SKIP：不丢弃任何匹配到的结果

2、SKIP_TO_NEXT： 保留第一个匹配到的结果

3、SKIP_PAST_LAST_EVENT： 保留最后一个匹配到的结果

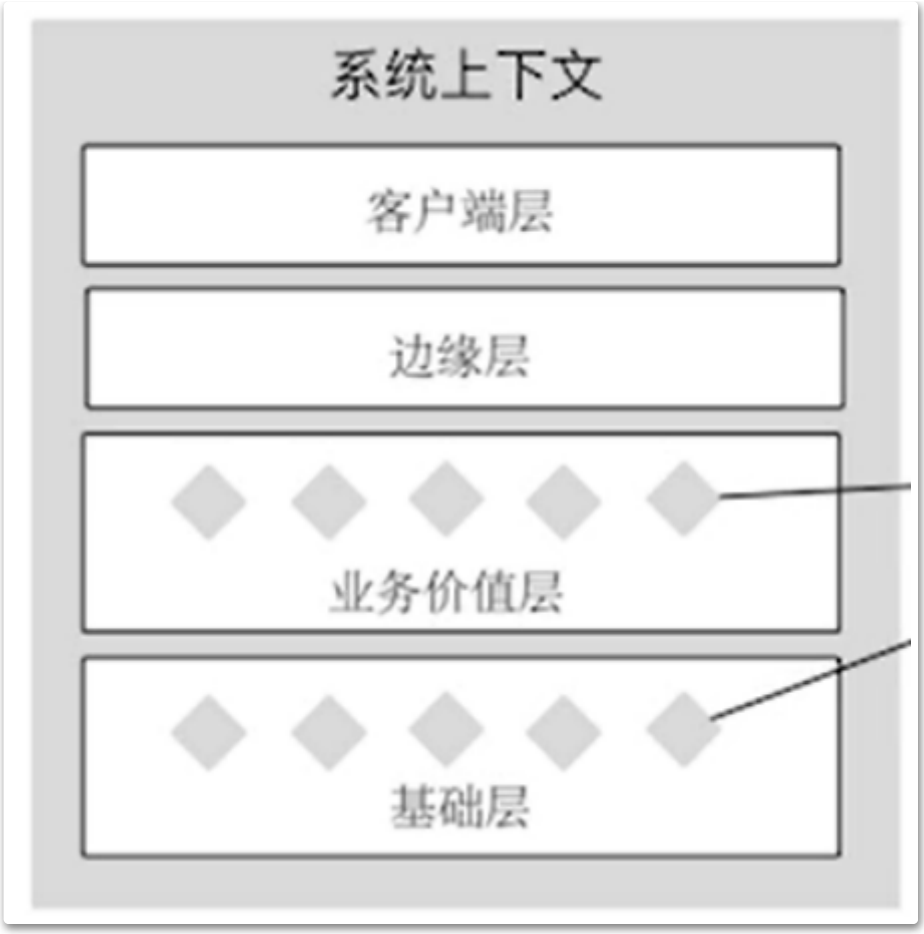
关于时间范围，within，一组匹配的结果必须在一定的时间范围内。

超时的匹配结果是会丢弃，但是也可以通过处理函数手动进行收集。

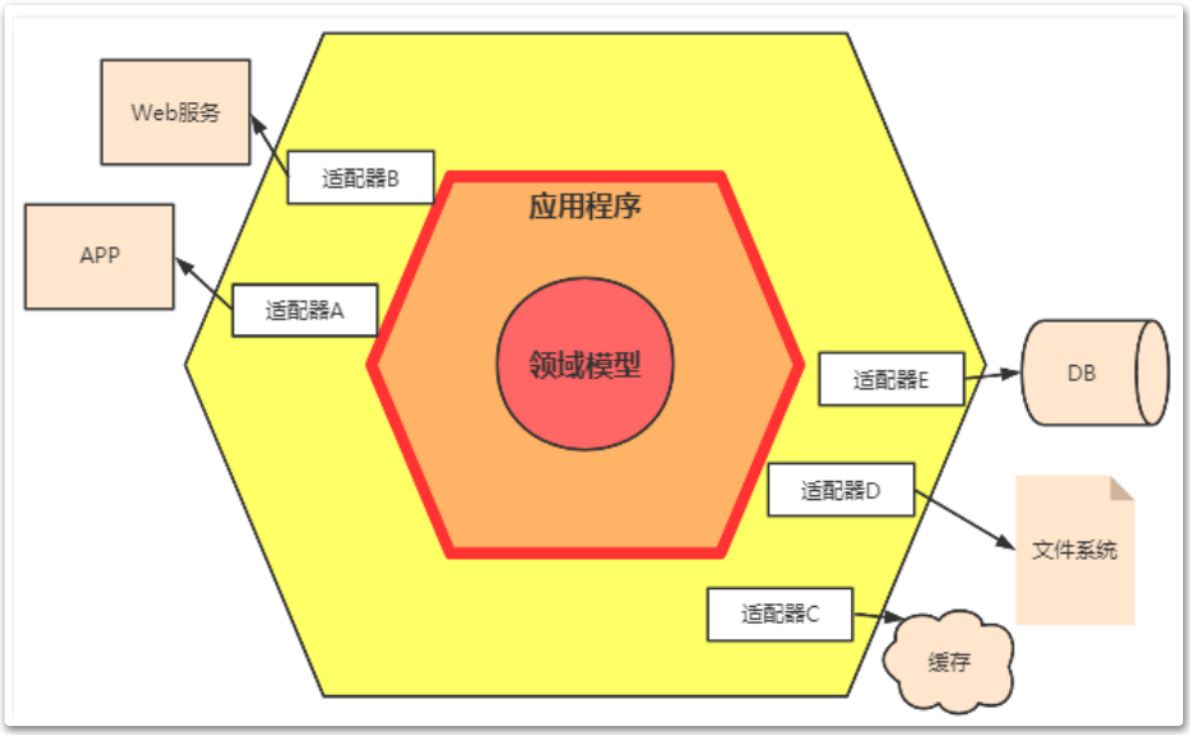
五、项目三高架构扩展分析

1、实时风控CQRS模式分析

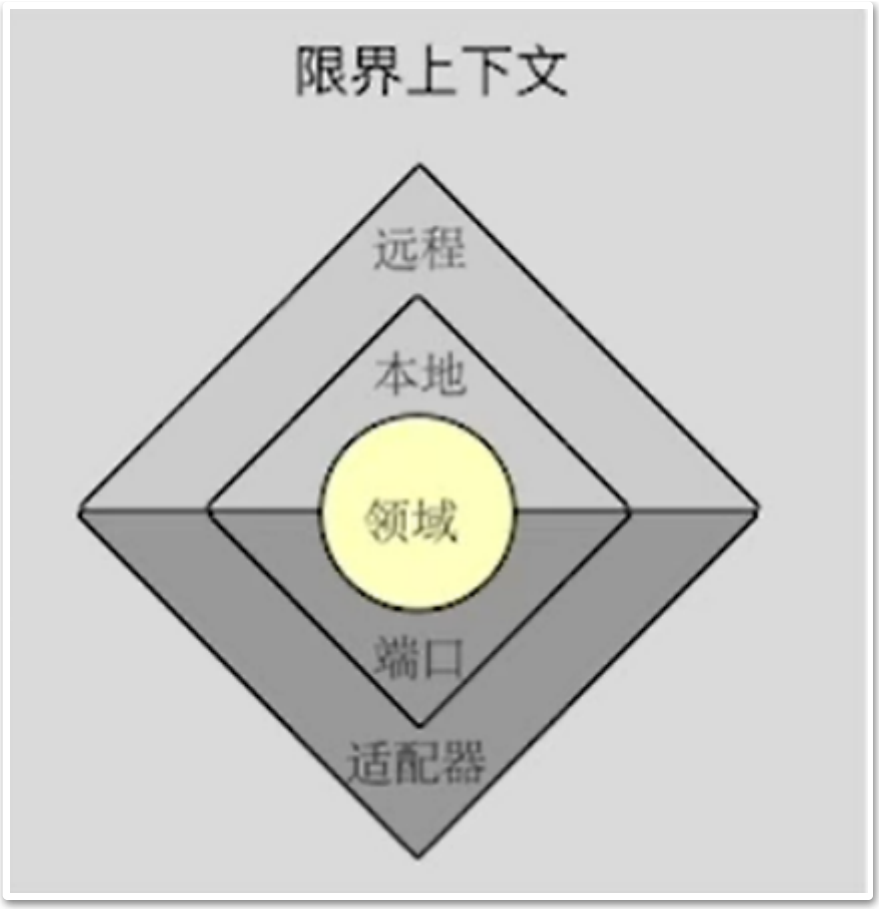
1、从整体来看，实时风控和非实时风控都会呈现出一个比较统一的整体结构。



2、从细节方面来看，将传统的MVC层次划分转换成为纵向的业务划分。



3、再来看基础层的统一服务处理。



4、最后还有一种畅想。就是将这些独立稳定的功能模块抽离出具体的项目，形成稳定的领域仓库。

2、预留的任务

BFF模块功能加强。

- 业务参数管理
- 实时风控的规则包获取以及更新。
- 实时风控日志追踪。 所有的实时风控处理消息都已经存入到了HBase，查询、分析、告警之类的功能。

行动起来，你终将收获自己的星辰大海。