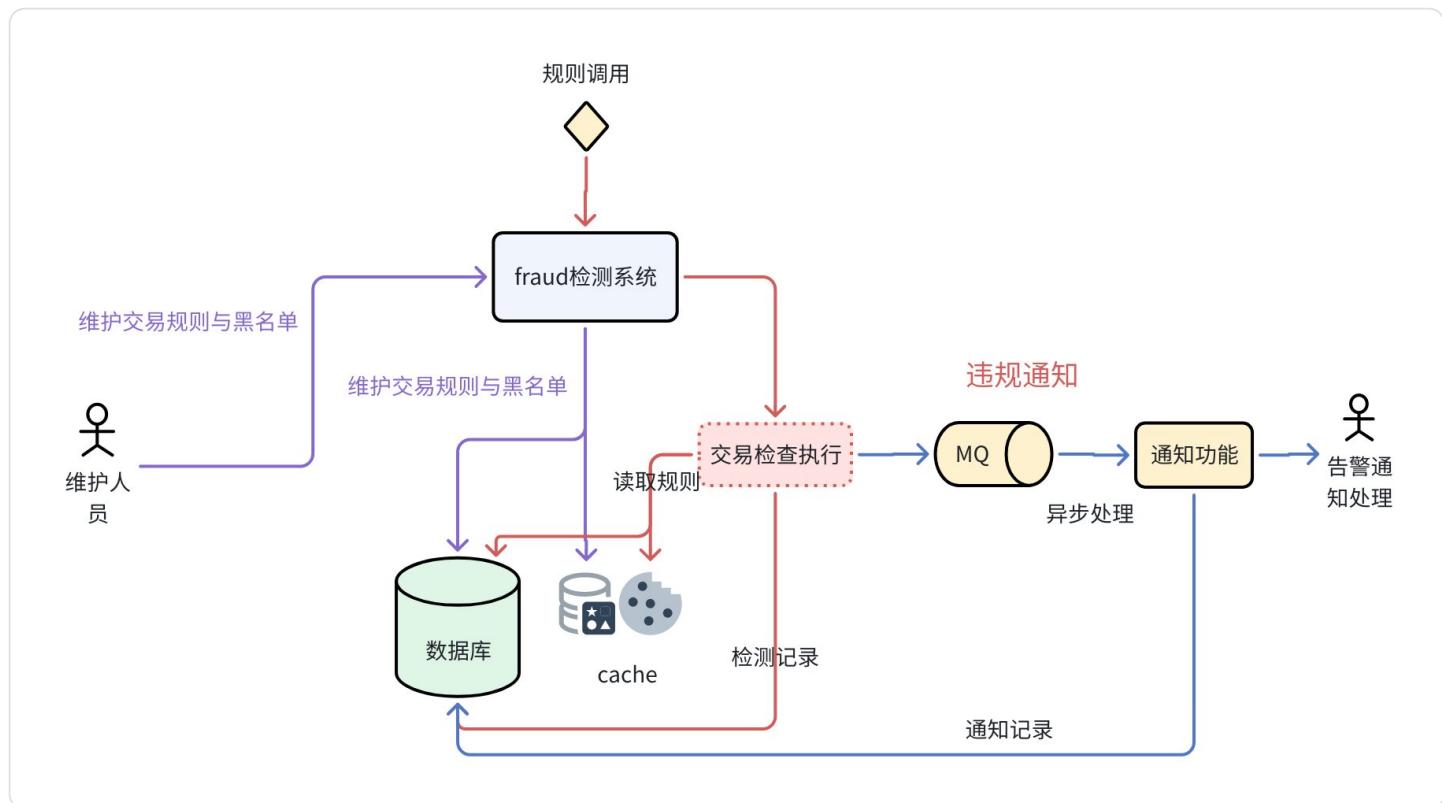


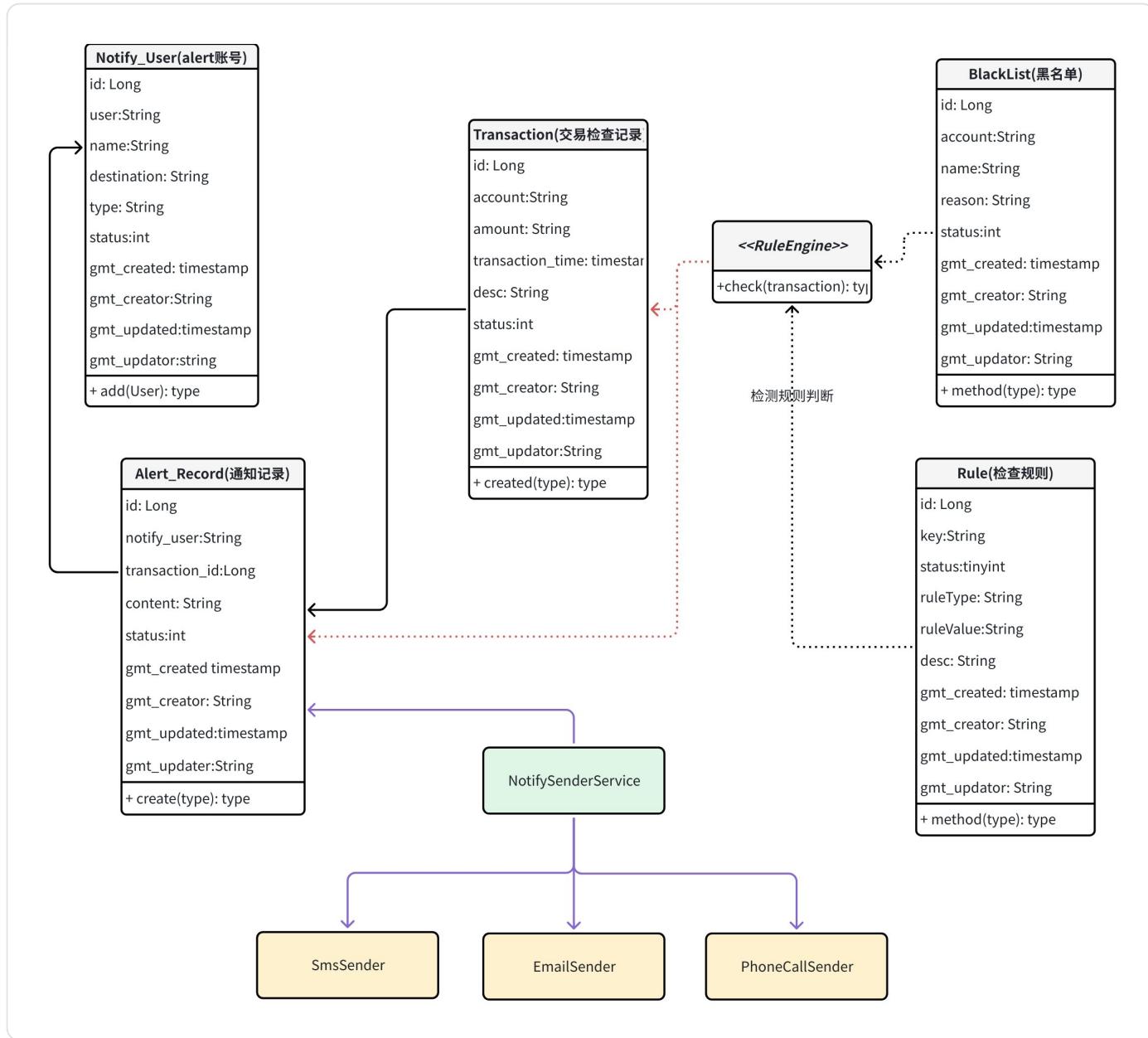
设计文档

一. 技术方案设计



二. 业务与数据模型

1. 业务模型



2. 数据模型

2.1 blacklist(黑名单记录表)

名称	类型	长度	是否为空	备注说明
id	long	20	不为空，可自动生成	黑名单id
account	varchar	64	不为空	黑名单对应的账户
name	varchar	64	不为空	黑名单对应的name
reason	varchar	2000	不为空	加入黑名单的原因
status	int	4	不为空， 默认0，	状态描述， 0， 正常 检查， 其他状态。

gmt_create_d	timestamp	timestamp	不为空， 默认当前写入数据时间	创建人
gmt_create_r	varchar	64	不为空	创建人
gmt_updated	timestamp	timestamp	不为空	更新时间
gmt_updater	varchar	64	不为空	更新人

2.2 rule(检查规则表)

名称	类型	长度	是否为空	备注说明
id	long	20	不为空， 可自动生成	检查规则ID
key	varchar	64	不为空	id对应的key, 如 amount_threshold,
rule_type	varchar	64	不为空	检查规则类型,amount blacklist time
value	varchar	64	不为空	检查规则值, 1000, 2-5, true
desc	varchar	512	可为空	描述
status	int	4	不为空， 默认0,	状态描述。0, 正常检查, 其他状态。
gmt_create_d	timestamp	timestamp	不为空， 默认当前写入数据时间	创建人
gmt_create_r	varchar	64	不为空	创建人
gmt_updated	timestamp	timestamp	不为空	更新时间
gmt_updater	varchar	64	不为空	更新人

2.3 transaction(交易检查记录表)

名称	类型	长度	是否为空	备注说明
id	long	20	不为空, 可自动生成	交易id
transaction_id	varchar	128	不为空	交易的原始ID
account	varchar	64	不为空	交易账户
amount	varchar	128	不为空	交易金额
transaction_time	timestamp		不为空	交易时间
desc	varchar	2000	可为空	交易备注说明
status	int	4	不为空, 默认0,	检测结果0:正常未触发规则, 不做通知只记录; 1不正常交易, 触发规则, 通知并且记录
reason	varcahr	2000	可为空,	触发规则出现的原因详细记录
gmt_created	timestamp	timestamp	不为空, 默认当前写入数据时间	创建黑名单人
gmt_creator	varchar	64	不为空	创建人
gmt_update_d	timestamp	timestamp	不为空	更新时间
gmt_updater	varchar	64	不为空	更新人

2.4 alert_record(告警通知记录表)

名称	类型	长度	是否为空	备注说明
id	long	20	不为空, 可自动生成	记录id
notify_user	varchar	64	不为空	通知到的人
destination	varchar	64	不为空	通知目标
notify_type	varchar	64	不为空	通知类型
	long	20	不为空	交易对应的ID

transaction_id				
amount	varchar	128	不为空	交易额度
content	varchar	2000	不为空	通知的内容
status	int	4	不为空, 默认0,	状态描述0,正常;其他状态。是否正常通知以及阅读
gmt_create_d	timestamp	timestamp	不为空, 默认当前写入数据时间	创建黑名单人
gmt_creato_r	varchar	64	不为空	创建人
gmt_update_d	timestamp	timestamp	不为空	更新时间
gmt_updat_e_r	varchar	64	不为空	更新人

2.5 notify_user(通知人员表)

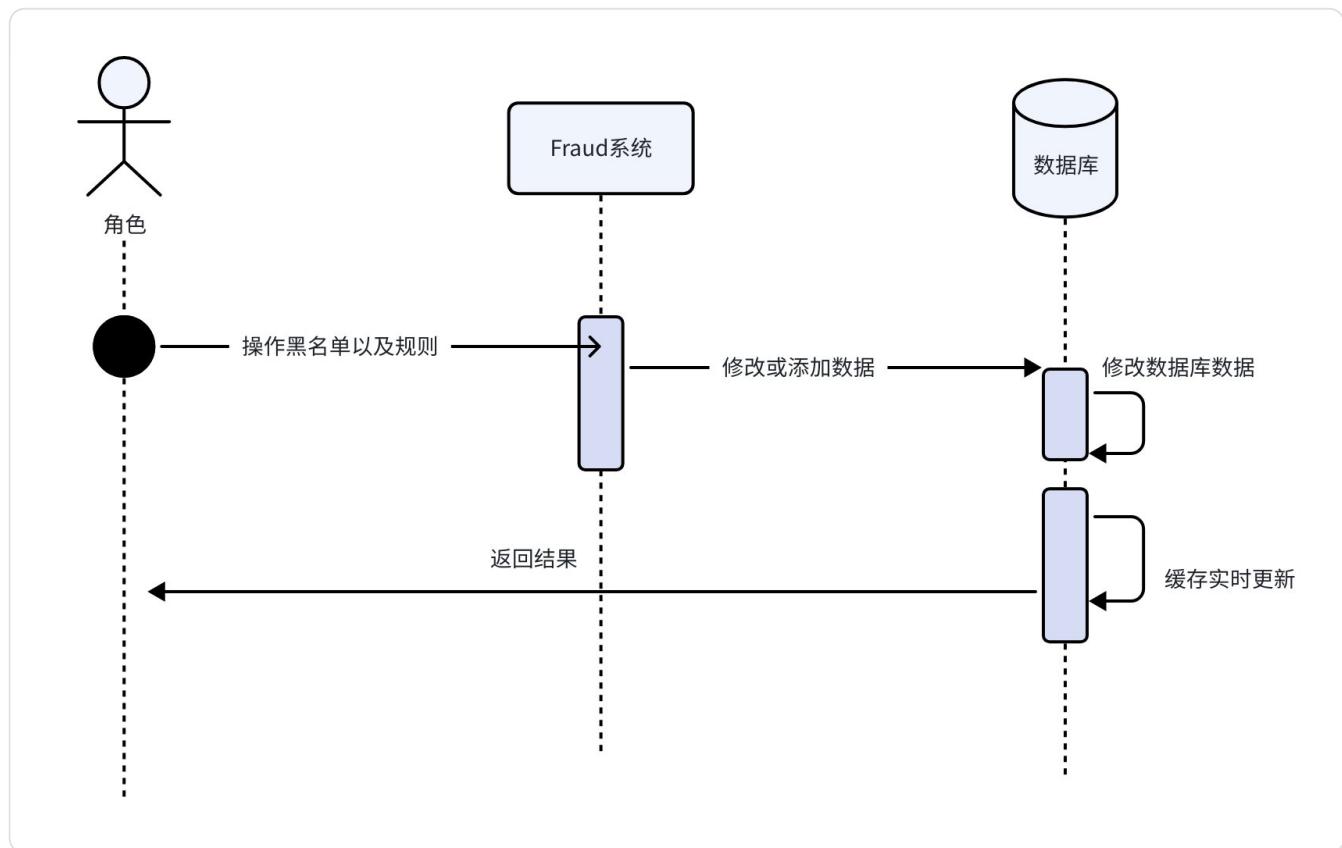
名称	类型	长度	是否为空	备注说明
id	long	20	不为空, 可自动生成	id
user	varchar	64	不为空	通知对应的用户
name	varchar	64	不为空	对应的name
destination	varchar	64	不为空	触达目标
type	varchar	16	不为空	通知类型: SMS短信; Email:邮件; Phone:电话等
status	int	4	不为空, 默认0,	状态描述, 0, 正常, 其他状态。
gmt_create_d	timestamp	timestamp	不为空, 默认当前写入数据时间	创建黑名单人
gmt_creato_r	varchar	64	不为空	创建人

gmt_updated	timestamp	timestamp	不为空	更新时间
gmt_updater	varchar	64	不为空	更新人

三. 业务流程

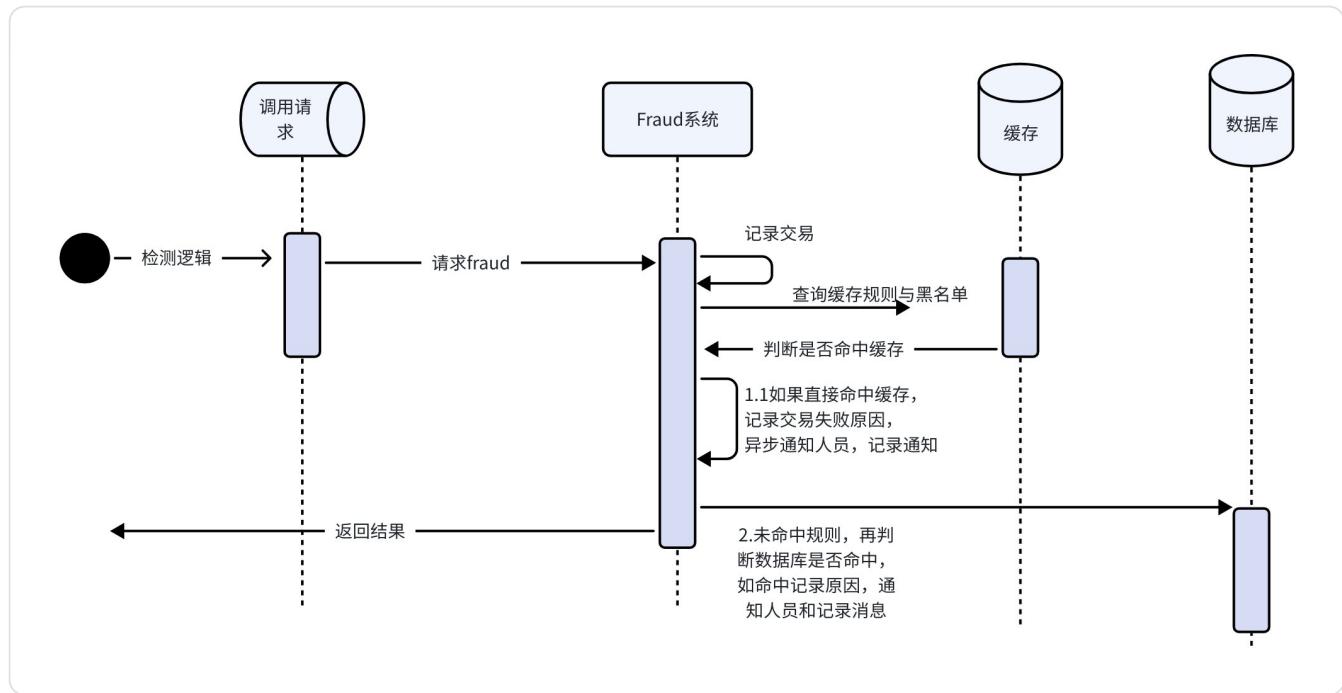
1. 维护数据相关业务流程

维护数据流程主要是规则修改，黑名单维护，查看交易数据，通知查看，维护通知人员信息等基本操作



2. 实时检测交易流程

实时检测交易主要是检查交易是否发生触发规则，发送通知，记录交易信息等内容



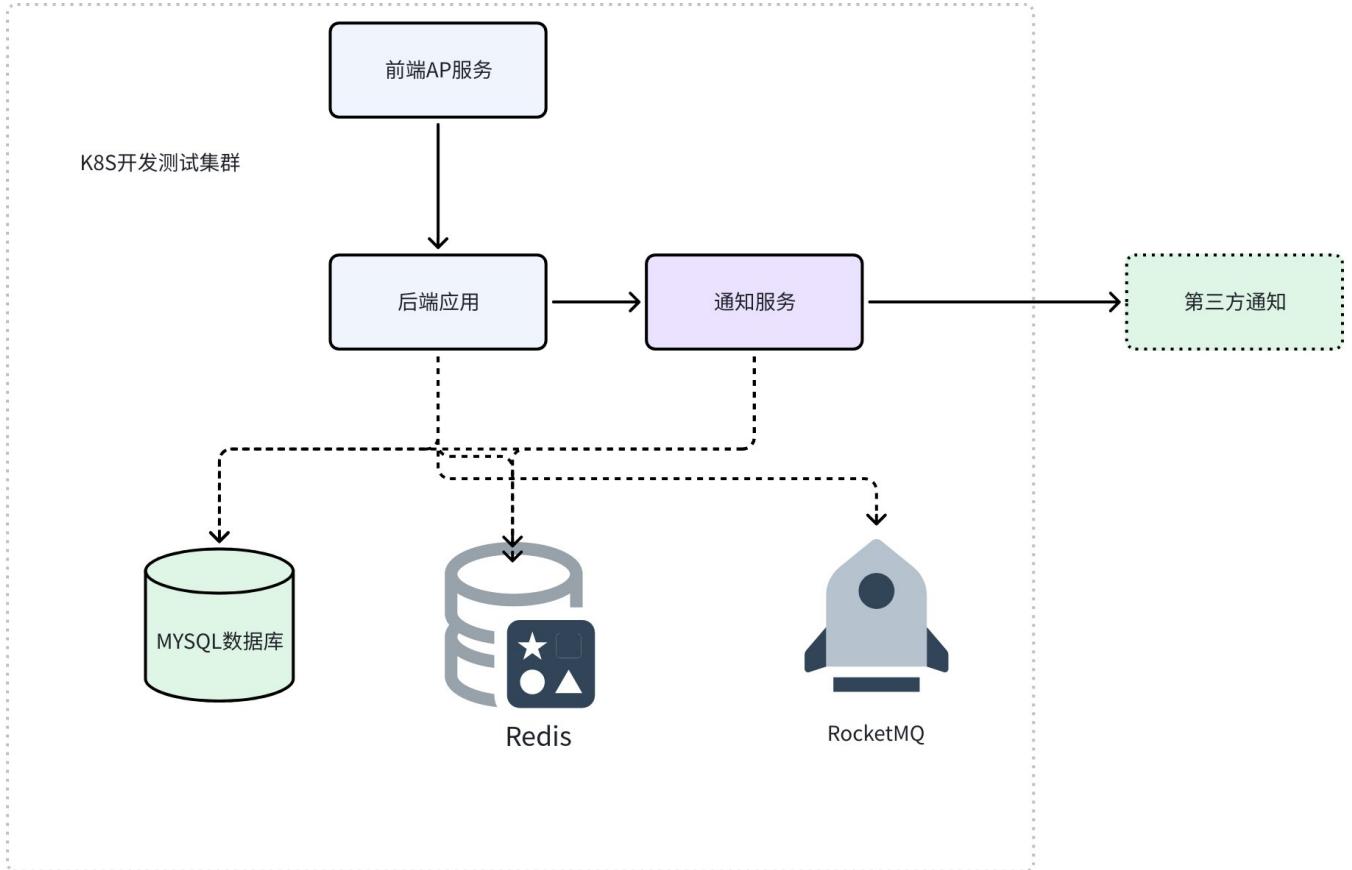
请求的并发幂等控制。

- i. 为了防止重复提交错误提交等，在每次请求http header头里面新增一个"Idempotent-Token"，唯一的uuid值，放入缓存key中，过期时间5秒钟，每次检查请求token是否存在，如果存在直接返回重复提交信息。
- ii. 数据库对于transaction id 进行唯一主键索引，避免脏数据的出现
- iii. 同时针对RPC/GRPC调用的逻辑可以使用分布式锁进行锁定即可，可以避免并发和重复的问题

四. 部署发布策略设计

1. 对于开发测试环境部署

我们可以将所有服务如AP应用，前端应用，MySQL，Redis，MQ等全部部署在K8S集群上面



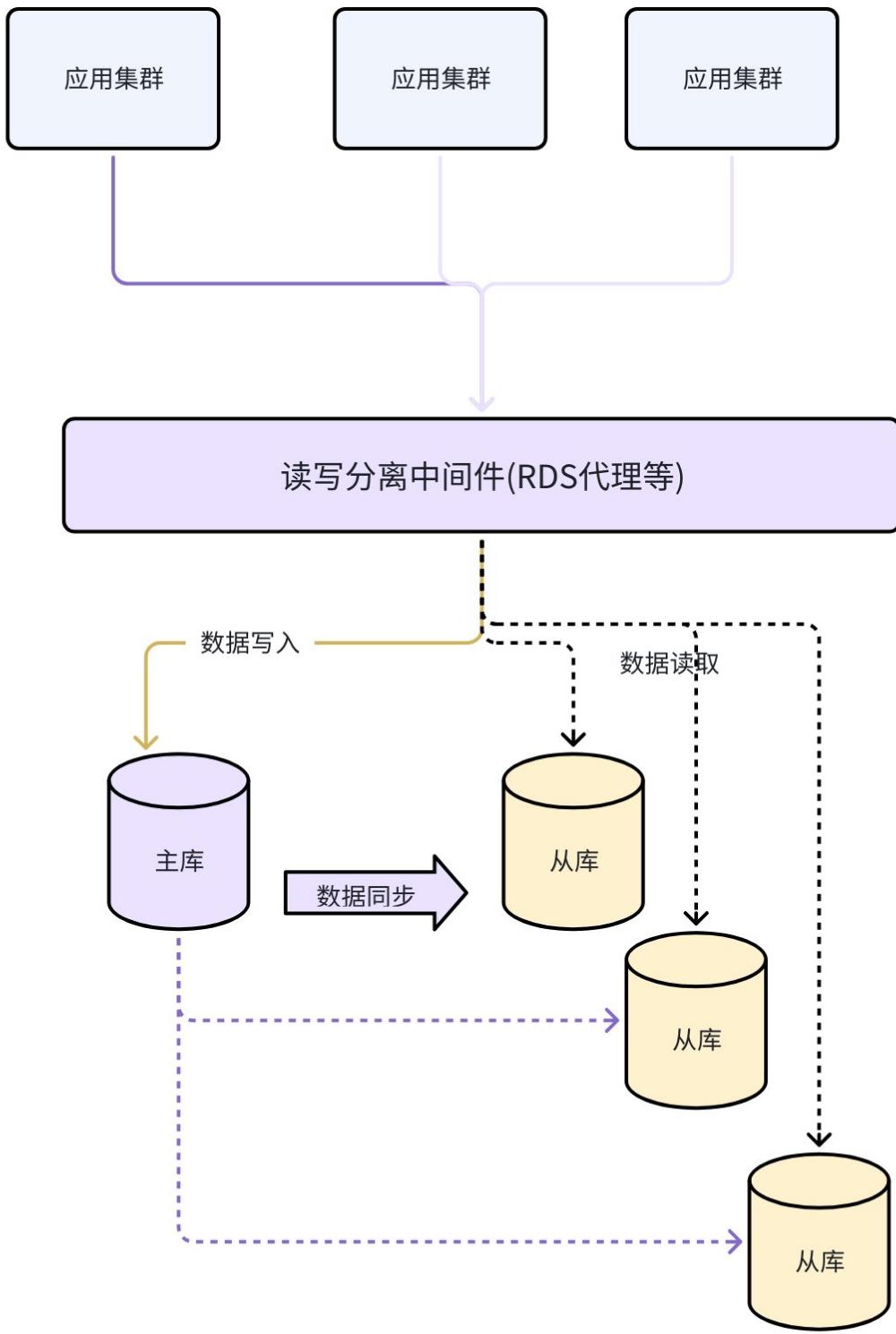
2. 生成环境扩展部署

对于生成环境而言，资源是相对比较多的，对于稳定性，容量等都有要求和资源。可以对整个应用进行扩展部署。

2.1 数据库的扩展

mysql可以扩展为主从读写模式，甚至是使用分布式数据库等。

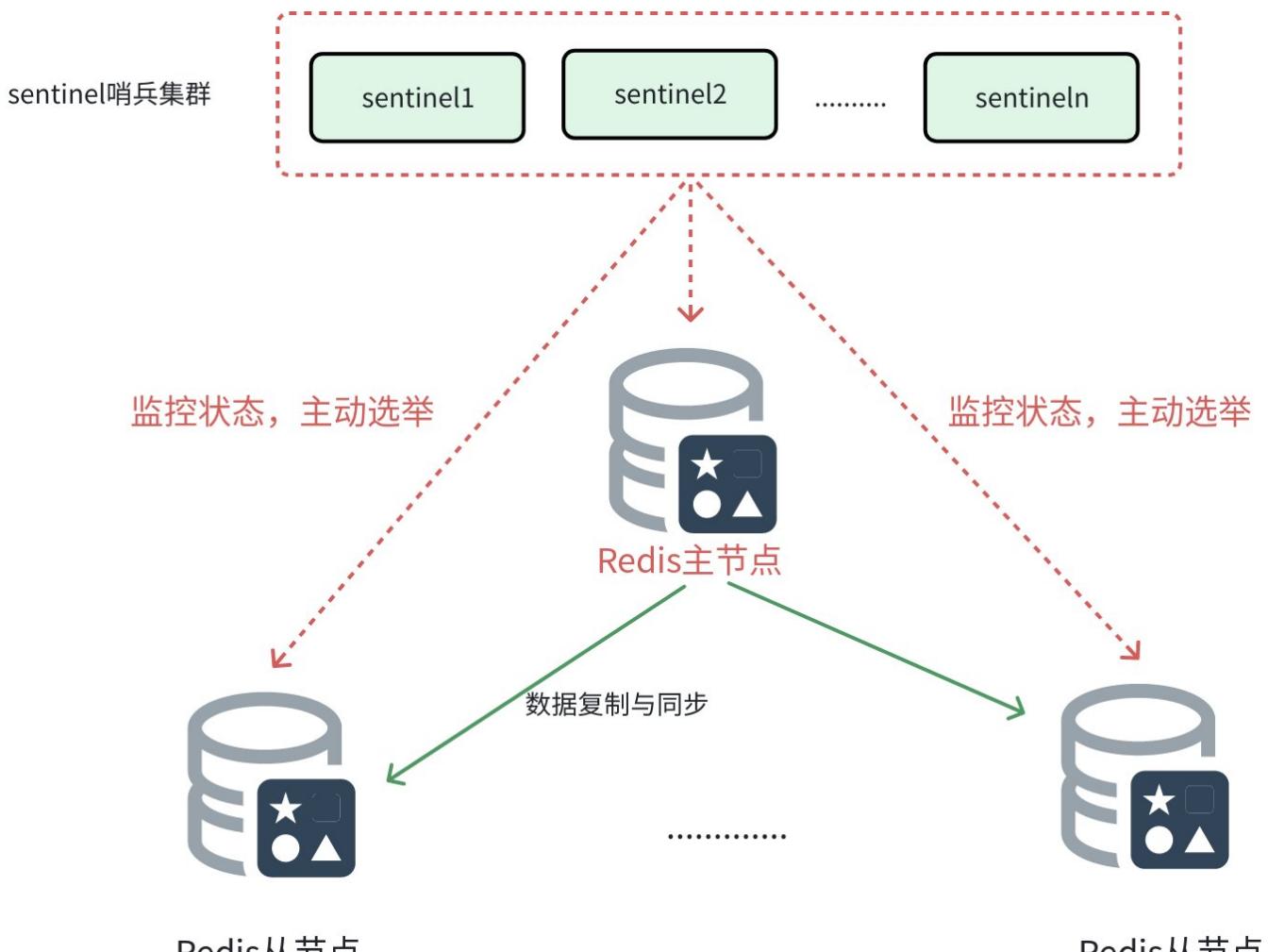
云服务本身的数据库使用很方便，如阿里云RDS，自带读写分离代理，以及分布式数据库服务等。对于超大容量以及OLAP/OLTP场景都有很方便的支持。对于业务不入侵，使用非常方便。



2.2 redis的扩展

同样，对于集中模式缓存redis中间件服务，我们也可以采用sentinel集群模式，保证高可用以及性能。也可以直接使用云服务的缓存服务，诸如阿里云的Tair，具有多种特性如可靠性、稳定性、弹性、安全、易用性等多个维度。避免维护搭建的成本。

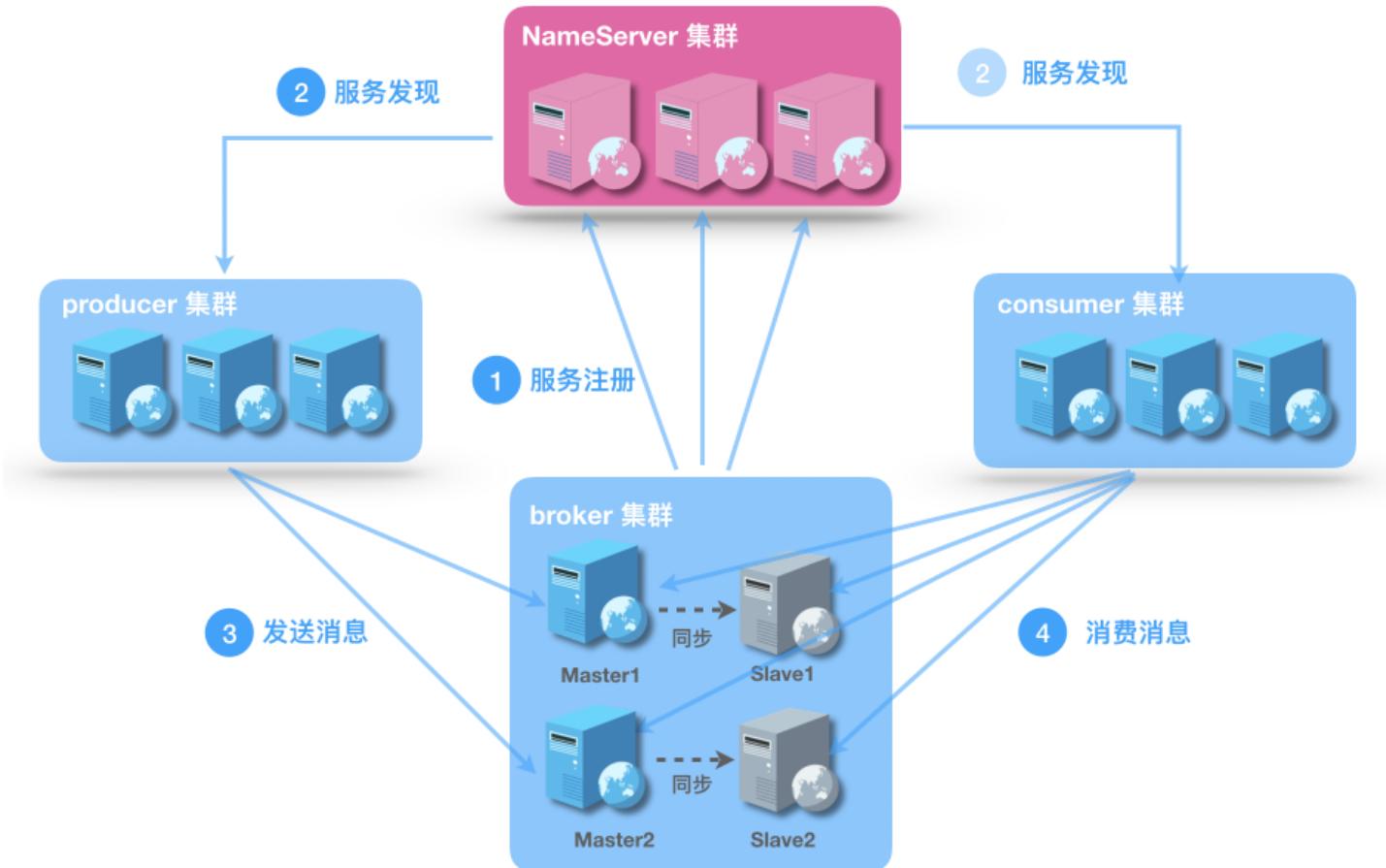
下面的内容用redis sentinel模式进行部署方案。同时使用集群模式也可以达到高可用的目的与标准



2.3 MQ的扩展

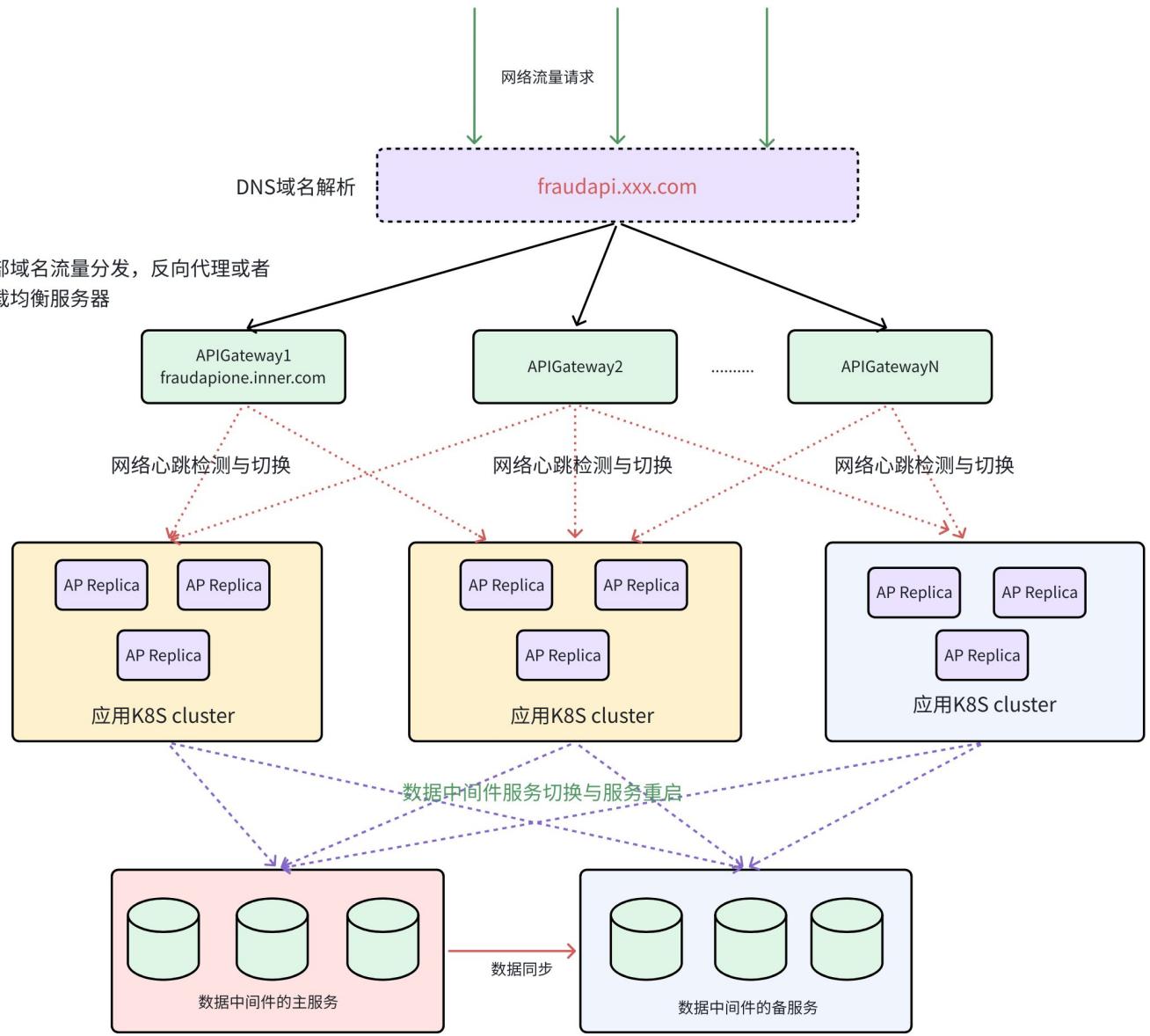
针对MQ的可用性而言，可以使用官方建议主从模式，当然也可以直接使用云服务的MQ消息服务。如阿里云ONS消息服务等，省去不少的运维成本以及搭建维护的成本。

如下面的划分：当然选择了可用性，必然会对性能有一定的影响，从而造成取舍问题。



2.4 生成环境AP应用的扩展

针对生产环境的应用可靠性而言。可以使用多集群部署模式，以及代理域名分发等机制。从而达到应用的高可用性。



五. 本地开发与部署测试

1. 工程介绍与说明

整体工程源码包括下面几个项，文件夹下面存放必要的文件信息。

1.1 deploy文件夹，

主要存放部署需要的到k8s集群环境的部署yaml文件。包括以下几个文件：

1. fraud-deploy.yaml。主要用于部署fraud应用，service，configmap等文件内容。
2. mysql-deploy.yaml、redis-deploy.yaml、rockermq-deploy.yaml文件。为部署应用的辅助组件，如果是已经有对应的环境，可以直接修改到对应的配置即可

1.2 dockers文件夹

1. 主要是存放dockerfile文件。用于制作fraud对应docker镜像

1.3 docks文件夹

1. 主要是设计相关的文档。包括相应的架构，截图，说明文件等。

1.4 libs文件夹

是本地运行程序需要的依赖包。在工程模块划分以及接口返回，日志规范，错误处理，API规范，依赖等方面进行了一层封装。在本地运行之前，需要将libs下面的文件sct.zip解压到对应的maven仓库，

主要是一下路径：\${maven}/m2/repository/com。正确的路径如下图所示：

```
..ace-sct/fraud (-zsh)      #1 ..shboard-2.0.0 (-zsh)      #2 root@iZbp10gx37urdwfgsln8w1Z: ~/dep
→ sct pwd
/Users/helloy/.m2/repository/com/sct  maven本地仓库路径
sct ls
clouds  commons  fraud  rocketmq  springboot
sct
必要的工程包文件
```

1.5 fraudfront 是前端应用

工程对应的前端工程应用。只是简单的前端展示。工程主要使用基于 Vue 3 + Vite 构建
本地编译使用逻辑可以按照前端Readme工程使用

1.6 sql文件夹

sql文件夹下面是schema.sql。用于初始化数据库的执行脚本以及SQL定义等。

1.7 应用工程模块如下图所示说明

fraud-api、fraud-biz、fraud-commons、fraud-repository、fraud-starter

从工程角度来讲，独立的 module负责对应的责任，确保本不同的功能在不同的模块下面，既可以清晰展示源代码逻辑，也可以防止大量的java原文件混淆交织。同时也利于业务的维护和变更。

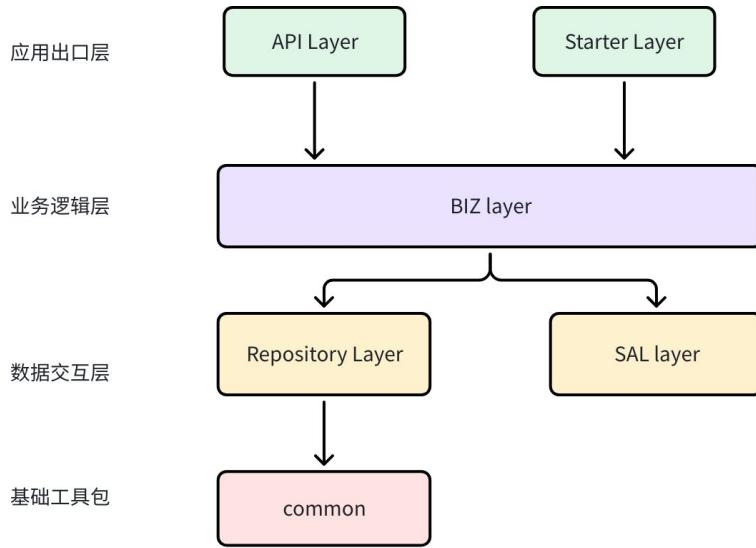
业务的出两个：api与starter，API用于内部服务之间的调用与接口规范，如GRPC/RPC/openfegin接口等。starter用于工程的启动配置，restfull相关操作接口等

业务的核心逻辑在biz下面，包括service，cache等服务层的接口以及逻辑类。

common是通用模块。主要包括工具类，枚举类型，错误码以及异常处理方法等内容。

repository是数据处理模块。包括数据模型DO对象，mapper接口，mapper.xml的文件。

如果有必要甚至可以包括调用外部模块SAL(service abstract layer)，message消息层等等。



2. 本地运行与接口测试

工程使用springboot3.2/spring/maven/mysql8/redis/rocketmq等。在本地启动前需要设置对应的环境信息。如安装MySQL/Redis/RocketMQ。

由于网络环境以及资源因素：local里面的数据库/redis等资源不一定能正常使用，可以略过测试直接启动。

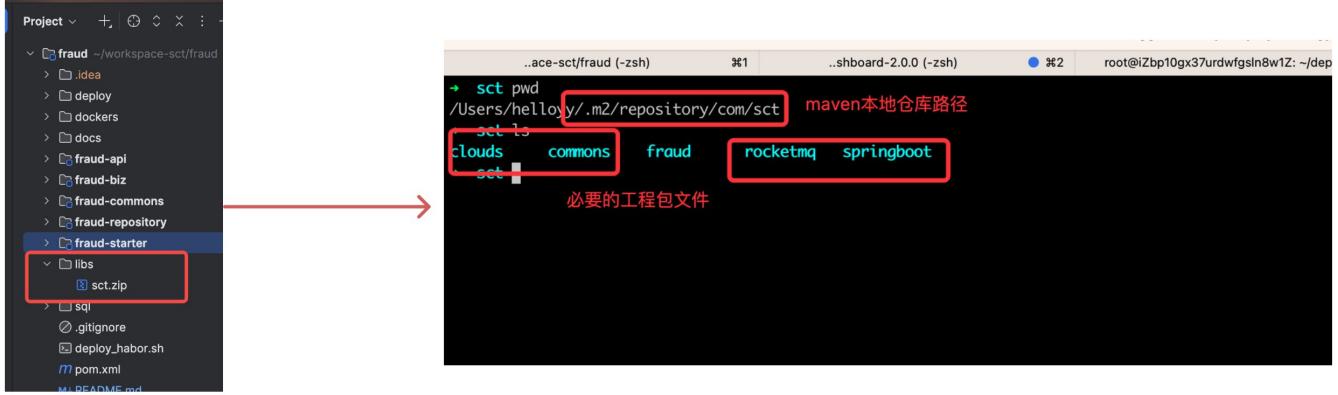
2.1 设置本地环境

1. 设置好本地JDK环境，目前使用JDK21版本
2. 安装设置mysql可以参考(<https://developer.aliyun.com/article/1039891>)
3. Redis安装可以参考(https://redis.io/docs/latest/operate/oss_and_stack/install/archive/install-redis/)
4. RocketMQ可以参考(<https://rocketmq.apache.org/zh/docs/quickStart/01quickstart>)

以上组件安装确认无误

2.2 拷贝依赖包到maven库

将libs下面的文件sct.zip解压到对应的maven仓库。如下图：



2.3 修改对应的配置文件

修改对应的配置文件(本工程为application.yaml)信息.

有三处地方地方需要修改。mysql信息、redis信息、rocketmq信息。如下图所示

mysql信息修改：

```

29 datasource:
30   app: MySQL连接信息需要修改本地设置内容
31     type: com.zaxxer.hikari.HikariDataSource
32     jdbc-url: ${spring.datasource.url:jdbc:mysql://172.16.31.197:31306/fraud}
33     driver-class-name: com.mysql.cj.jdbc.Driver
34     username : ${spring.datasource.username:root}
35     password : ${spring.datasource.password:fraud123456}
36     hikari:
37       allow-pool-suspension: false
38       auto-commit: true

```

redis信息修改

```

max-request-size: 5120000B
data:
  redis:
    host: ${spring.redis.host:172.16.31.197}
    port: ${spring.redis.port:31379}
    database: 0
    password: ${spring.redis.password:changeme1234}
    client-type: jedis
    jedis:

```

rocketmq需要修改：

```

rocketmq:
  #name-server: ${rocketmq.namesrv.addr:172.16.31.197:30976}
  name-server: ${rocketmq.namesrv.addr:172.16.31.99:9876}

  producer:
    group: fraud-producer-group

  consumer:

```

2.4 启动工程

1-2-3步骤完成之后可以进行工程编译以及启动，当没有出现任何错误信息的时候启动成功，如下图所示：

```

Projectfraudmaster
application.yml
rocketmq:
  name-server: ${rocketmq.namesrv.addr:172.16.31.197:30976}
  producer:
    group: fraud-producer-group
  consumer:

Debug FraudApplication
Threads & Variables Console Beans Health Mappings Environment
2025-06-22 16:05:29.836 [main] [DEBUG] Mappings - o.s.b.a.w.e.BasicErrorHandler: { [/error]: error(HttpServletRequest)
  { [/error], produces [text/html]}: errorHtml(HttpServletRequest,HttpServletResponse)
2025-06-22 16:05:29.957 [main] [DEBUG] Mappings - 'beanNameHandlerMapping' {}
2025-06-22 16:05:30.801 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@60193e6c
2025-06-22 16:05:30.833 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - After adding stats (total=6, active=0, idle=6, waiting=0)
2025-06-22 16:05:31.597 [main] [DEBUG] InternalLoggerFactory - Using SLF4J as the default logging framework
2025-06-22 16:05:31.746 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@40000000
2025-06-22 16:05:31.780 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - After adding stats (total=7, active=0, idle=7, waiting=0)
2025-06-22 16:05:33.268 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@581d96
2025-06-22 16:05:33.294 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - After adding stats (total=8, active=0, idle=8, waiting=0)
2025-06-22 16:05:34.614 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@50b4ef57
2025-06-22 16:05:34.647 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - After adding stats (total=9, active=0, idle=9, waiting=0)
2025-06-22 16:05:35.402 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@7c28230c
2025-06-22 16:05:35.436 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - After adding stats (total=10, active=0, idle=10, waiting=0)
2025-06-22 16:05:35.436 [HikariPool-1 connection adder] [DEBUG] HikariPool - HikariPool-1 - Connection not added, stats (total=10, active=0, idle=10, waiting=0)
2025-06-22 16:05:37.024 [main] [INFO] FraudApplication - Started FraudApplication in 17.544 seconds (process running for 19.434)
2025-06-22 16:05:56.404 [HikariPool-1 housekeeper] [DEBUG] HikariPool - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
2025-06-22 16:05:56.404 [HikariPool-1 housekeeper] [DEBUG] HikariPool - HikariPool-1 - Fill pool skipped, pool has sufficient level or currently being filled.
2025-06-22 16:06:26.408 [HikariPool-1 housekeeper] [DEBUG] HikariPool - HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0)
2025-06-22 16:06:26.408 [HikariPool-1 housekeeper] [DEBUG] HikariPool - HikariPool-1 - Fill pool skipped, pool has sufficient level or currently being filled.

fraud > fraud-starter > src > main > resources > application.yml
84:12 LF UTF-8 2 spaces

```

2.5 接口测试

接口文档放在doc目录下面的fraud-local.postman_collection.json文件。可以导入到本地post即可进行测试验证。

2.5.1 创建和修改检查规则测试

HTTP fraud-local / rule / 更新检查规则

fraud
fraud-local
rule
blacklist
alertRecord

POST 创建检测规则
PUT 更新检查规则
GET 检测规则详情
POST 创建黑名单
PUT 更新黑名单
GET 查询详情信息
GET 分页查询黑名单信息
DEL 删除黑名单
alertRecord
GET 查询通知详情信息
DEL 删除通知记录

PUT http://localhost:8080/fraud/rule/update

Params Authorization Headers (9) Body Scripts Tests Settings

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results

{ } JSON ▾ Preview Visualize

```
1 {  
2   "code": "SUCCESS",  
3   "message": "操作成功",  
4   "data": true  
5 }
```

2.5.2 检查黑名单接口测试

fraud
fraud-local
rule
blacklist
alertRecord
nofityUser
transactionRecord
POST 实时交易反欺诈检测
larkapi
告警功能

POST 创建黑名单
PUT 更新黑名单
GET 查询详情信息
GET 分页查询黑名单信息
DEL 删除黑名单

HTTP fraud-local / blacklist / 创建黑名单

POST http://localhost:8080/fraud/blacklist/create

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON ▾

{ } JSON ▾ Preview Visualize

```
1 {  
2   "account": "zhags1n1322",  
3   "name": "zsdha12g1",  
4   "reason": "异常交易1信息港1"  
5 }
```

Body Cookies Headers (5) Test Results 200

{ } JSON ▾ Preview Visualize

```
1 {  
2   "code": "2003",  
3   "message": "数据已经存在"  
4 }
```

2.5.3 反欺诈检测接口测试

```

{
    "transactionId": "228u48339stds",
    "account": "ew185r4",
    "amount": "45659666",
    "transactionTime": 1750477432000,
    "description": "交易信息454554"
}

```

```

{
    "code": "SUCCESS",
    "message": "操作成功",
    "data": {
        "checkResult": true,
        "reasons": "检测交易信息:\n {\\"transactionId\\":\\"228u48339stds\\",\\"account\\":\\"ew185r4\\",\\"amount\\":\\"45659666\\",\\"transactionTime\\":1750477432000,\\"status\\":1}\nTransaction amount exceeds: 70000.0"
    }
}

```

```

sageDTO[account=ew1478r4,amount=45659666,content=交易信息: {"transactionId":"228u48764539stds","account":"ew1478r4","amount":"45659666","transactionTime":1750477432000,"status":1}触发反欺诈规则
触发原因如下: ["Transaction amount exceeds: 70000.0"],destination=18624343424,gmtCreator=<null>,gmtUpdater=<null>,notifyUser=zhang32,traceId=<null>,transactionId=228u48764539stds,type=phone]
2025-06-22 16:30:59.712 [http-nio-8080-exec-1] [INFO ] AlertRecordMessageSendProducerImpl - AlertRecordMessageSendProducerImpl,send alert to fraudAlertTopic, alert:AlertRecordMessageSendProducerImpl[account=ew1478r4,amount=45659666,content=交易信息: {"transactionId":"228u48764539stds","account":"ew1478r4","amount":"45659666","transactionTime":1750477432000,"status":1}触发反欺诈规则
触发原因如下: ["Transaction amount exceeds: 70000.0"],destination=18624343424,gmtCreator=<null>,gmtUpdater=<null>,notifyUser=zhang342,traceId=<null>,transactionId=228u48764539stds,type=phone]
2025-06-22 16:30:59.715 [http-nio-8080-exec-1] [INFO ] AlertRecordMessageSendProducerImpl - AlertRecordMessageSendProducerImpl,send alert to fraudAlertTopic, alert:AlertRecordMessageSendProducerImpl[account=ew1478r4,amount=45659666,content=交易信息: {"transactionId":"228u48764539stds","account":"ew1478r4","amount":"45659666","transactionTime":1750477432000,"status":1}触发反欺诈规则
触发原因如下: ["Transaction amount exceeds: 70000.0"],destination=186243333424,gmtCreator=<null>,gmtUpdater=<null>,notifyUser=zhang3342,traceId=<null>,transactionId=228u48764539stds,type=phone]
2025-06-22 16:30:59.718 [http-nio-8080-exec-1] [INFO ] TXEvalEngine - TXEvalEngine evaluation completed, transaction pass
2025-06-22 16:30:59.887 [ConsumeMessageThread_fraud_consumer_2] [INFO ] AlertRecordMessageListener - AlertRecordMessageListener.onMessage:[{"account":"ew1478r4","amount":"45659666","content":"交易信息: {\\"transactionId\\":\\"228u48764539stds\\",\\"account\\":\\"ew1478r4\\",\\"amount\\":\\"45659666\\",\\"transactionTime\\":1750477432000,\\"status\\":1}触发反欺诈规则\n触发原因如下: [\\"Transaction amount exceeds: 70000.0\\"]","destination":"18624343424","notifyUser":"zhang342","transactionId":"228u48764539stds","type":"phone"}]
2025-06-22 16:30:59.887 [ConsumeMessageThread_fraud_consumer_3] [INFO ] AlertRecordMessageListener - AlertRecordMessageListener.onMessage:[{"account":"ew1478r4","amount":"45659666","content":"交易信息: {\\"transactionId\\":\\"228u48764539stds\\",\\"account\\":\\"ew1478r4\\",\\"amount\\":\\"45659666\\",\\"transactionTime\\":1750477432000,\\"status\\":1}触发反欺诈规则\n触发原因如下: [\\"Transaction amount exceeds: 70000.0\\"]","destination":"186243333424","notifyUser":"zhang3342","transactionId":"228u48764539stds","type":"phone"}]
2025-06-22 16:30:59.887 [ConsumeMessageThread_fraud_consumer_1] [INFO ] AlertRecordMessageListener - AlertRecordMessageListener.onMessage:[{"account":"ew1478r4","amount":"45659666","content":"交易信息: {\\"transactionId\\":\\"228u48764539stds\\",\\"account\\":\\"ew1478r4\\",\\"amount\\":\\"45659666\\",\\"transactionTime\\":1750477432000,\\"status\\":1}触发反欺诈规则\n触发原因如下: [\\"Transaction amount exceeds: 70000.0\\"]","destination":"18624343424","notifyUser":"zhang32","transactionId":"228u48764539stds","type":"phone"}]
2025-06-22 16:30:59.896 [ConsumeMessageThread_fraud_consumer_2] [INFO ] PhoneSender - PhoneSender: callPhone, phone=18624343424, content=交易信息: {"transactionId":"228u48764539stds","account":"ew1478r4","amount":"45659666","transactionTime":1750477432000,"status":1}触发反欺诈规则触发检查规则后会发送通知消息, 将详细的内容通知到某个人, 也会记录日志
触发原因如下: ["Transaction amount exceeds: 70000.0"]
2025-06-22 16:30:59.896 [ConsumeMessageThread_fraud_consumer_2] [INFO ] NotifySenderService - phoneSender,destination:18624343424
2025-06-22 16:30:59.896 [ConsumeMessageThread_fraud_consumer_1] [INFO ] PhoneSender - PhoneSender: callPhone, phone=18624343424, content=交易信息: {"transactionId":"228u48764539stds","account":"ew1478r4","amount":"45659666","transactionTime":1750477432000,"status":1}触发反欺诈规则
触发原因如下: ["Transaction amount exceeds: 70000.0"]
2025-06-22 16:30:59.901 [ConsumeMessageThread_fraud_consumer_1] [INFO ] NotifySenderService - phoneSender,destination:18624343424
2025-06-22 16:30:59.901 [ConsumeMessageThread_fraud_consumer_3] [INFO ] NotifySenderService - phoneSender,destination:186243333424

```

3. 本地开发代码说明

上面的2中已经说明了相关信息。下面是代码层面的说明与介绍

3.1 统一接口处理模式

接口经过封装基础组件的封装。返回统一的格式，只需要在controller层面返回Object对象即可。这样每个团队，前后端的格式进行统一。

接口错误码的定义包含：错误码，Debug信息(用于前后端调试)，alert信息(用于展示给用户)

统一在RPC/GRPC/Openfeign层面页 进行了必要的封装，避免不同不同服务团队之间的接口差异等。

```
73
74     @GetMapping("/detail")  ↳ zhouyang
75     public Object detailAlertDO(HttpServletRequest request,@Param("id") Long id){
76
77         logger.info("AlertRecordRestController.detailAlertDO, id={}", id);
78
79         if(ObjectUtil.isNull(id)){
80             throw new BusinessException(FraudRestresultCode.PARAMETER_ERROR,"参数不能为空");
81         }
82
83         AlertRecordDO result = this.alertRecordService.findAlertRecordById(id);
84         return result;
```

The screenshot shows a Postman request response. The status is 200 OK, with a response time of 109 ms and a size of 445 B. The response body is a JSON object:

```
1  {
2      "transactionId": "228u48764d39stds",
3
4
5
6
7
8 }
```

Annotations on the JSON response:

- "code": "SUCCESS", **错误码**
- "message": "操作成功", **alert message**
- "data": {
 "checkResult": true,
 "reasons": "检测交易信息:\n {"transactionId": "228u48764d39stds", "account": "ew14d78r4",
 \"transactionTime\":1750477432000, \"status\":1}\n Transaction amount exceeds: 70000.0"
}, **具体的 数据data**

3.2 统一的异常信息处理模式

在工程代码中有各种各样的异常，错误信息等，堆栈信息。通常情况下是不给前端用户的。针对这样的场景，封装了BusinessException业务异常类，配合统一的错误码，即可在团队之间，软件与用户之间尽量协调一致的功能，同时还为错误异常收集，分析等提供了统一的格式。

```

}
return vo;
} catch (Exception ex) {                                业务异常的地方抛出 Bus异常则基础框架会自动处理返回结果
    logger.info("fraudTxEvaluate,transaction={},error={}", txDO, ex.getMessage());
    if(ex instanceof DuplicateKeyException){
        throw new BusinessException(FraudRestresultCode.DATA_DUPLICATION_ERROR);
    }
    throw new BusinessException(FraudRestresultCode.ERROR,ex.getMessage());
}
```

3.3 各项配置文件说明：

以下几个配置分别对应的功能说明。local配置了可以在直接启动的环境信息。

application.yaml: 主要是本地测试开发以及默认的配置。

application-local.yaml: 主要是本地测试开发以及默认的配置。**在本地编译可以启用这个即可**

application-dev.yaml: 部署测试到k8s开发环境的配置文件。是自己测试用的可以不用管

一下两个是部署可能会用到的：

application-k8sall.yaml: k8s与allinone一致，指的是mysql/redis/rocketmq等集中在一个k8s集群的测试环境

application-k8sone.yaml: 应用k8s与mysql/redis/rocketmq分离管理

```
5   management
6   info:
7   env:
8   endpoints:
9   web:
10  examples:
11  endpoints:
12  health:
13  security:
14  endpoints:
15  health:
16  properties:
17  bootstrap.yml
18  logback-spring.xml
19  logback-spring.xml.back
20
21  spring:
```

fraud-starter

- src
- main
- java
- resources
 - properties
 - application.yml
 - application-dev.yml
 - application-k8sall.yml
 - application-k8sone.yml
 - application-local.yml
 - bootstrap.yml
 - </> logback-spring.xml
 - logback-spring.xml.back
- test

4. 前端工程本地启动

- 前端工程在fraudfront目录下面。主要使用的基础框架为Nodejs, vue3等。工程对应的前端工程应用。只是简单的前端展示。工程主要使用基于 Vue 3 + Vite 构建。本地编译使用逻辑可以按照前端 Readme工程使用。

代码块

```
1 ## 🚀 快速开始
2
3 ### 安装依赖
4 ``
5 npm install
6 ``
```

```
7 ## 启动本地开发服务
8 ````
9 npm run dev
10 ````
11 默认地址: http://localhost:5173
12
13 ## 构建生产环境
14 ````
15 npm run build
16 ````
```

六. 线上测试环境部署

1. 制作镜像文件

1.1 后端制作镜像文件

打包之前确认是否已经存在基础镜像: **openjdk:21**, 有时候网络问题无法正常下载这个官方镜像。需要提前下载

后端 dockerfile 位于工程的 dockers 下面。根据以上 5.2 的步骤完成本地编译或者启动后, 直接执行如下命令可以进行打包:

代码块

```
1 mvn clean install -Dmaven.test.skip=true
```

编译完成之后将 fraud-starter/target 目录下面的 fraud.jar 拷贝到 dockers 目录下面即可打包

代码块

```
1 cp fraud-starter/target/fraud.jar dockers/
```

打包镜像命令如下, 并且根据自己 harbor 或者其他镜像服务打 tag 即可使用。

代码块

```
1 cd dockers
2 docker build -t "fraud:v2" .
3 docker tag fraud:v2 crpi-1nz5mrV3oba11j6z.cn-
hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2
4 #根据实际推送到具体的仓库
5 docker push crpi-1nz5mrV3oba11j6z.cn-
hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2
```

以上打包的镜像在后面的k8s部署的时候需要使用。

crpi-1nz5mrv3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2

```
git clone https://github.com/taozhengtian/fraudapp.git  
cd fraudapp  
git checkout master  
git pull  
ls  
cp fraud-starter/target/fraud.jar dockers  
cd dockers  
ls  
Dockerfile fraud.jar  
docker build -t "fraud:v2".  
[+] Building 61.7s (9/9) FINISHED  
=> [internal] load build definition from Dockerfile  
=> [internal] load metadata for docker.io/fraud:v2
```

```
2 warnings found (use docker --debug to expand):  
- MaintainerDeprecated: Maintainer instruction is deprecated in favor of using label (line 8)  
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 3)
```

What's next:

```
View a summary of image vulnerabilities and recommendations: docker scan quickview  
dockers git:(master) ✘ docker tag fraud:v2 crpi-1nz5mrv3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2  
dockers git:(master) ✘ docker login --username=woolunn2012 crpi-1nz5mrv3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com  
Password:  
Login Succeeded  
dockers git:(master) ✘ docker push crpi-1nz5mrv3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2  
The push refers to repository [crpi-1nz5mrv3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp]  
a1ed1ceb84ed: Pushed  
d864ad9226fd: Pushed  
57118e83e915: Pushed  
10359c5dc4ba: Pushed  
601b48657e0c: Pushed  
b42107e74152: Pushed  
v2: digest: sha256:5555dc9c36cc9386d1704e5a313028d6d9ff7ff8fdf82df7aebc720bcf6ea0ec size: 1581
```

1.2 前端工程制作镜像文件

前端工程下面包含对应的dockerfile文件，直接在工程目录下执行如下打包命令即可。

Docker tag后推送到需要镜像推送到。

注意不同环境需要修改nginx.conf的配置。默认是k8s环境的。

代码块

```
1 docker build -t fraudfront:latest .  
2 docker image tag "fraudfront:latest"  
"harbor.streamcomputing.com/lark/fraudfraud:v1"
```

```

→ fraudfront git:(master) ✘ pwd
/Users/helloyy/workspace-sct/fraud/fraudfront
→ fraudfront git:(master) ✘ docker build -t fraudfront:latest .
[+] Building 5.0s (5/14)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 783B

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/mxs26c8jz1kaari5pb5ft8tgw
tag

What's next:
  View a summary of image vulnerabilities and recommendations: docker scan quickview
→ fraudfront git:(master) ✘ docker tag fraudfront:latest crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudfront:v2
→ fraudfront git:(master) ✘ docker push fraudfront:latest crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudfront:v2
docker: 'docker push' requires 1 argument

Usage: docker push [OPTIONS] NAME[:TAG] push

Run 'docker push --help' for more information
→ fraudfront git:(master) ✘ docker push crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudfront:v2
The push refers to repository [crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudfront]
8b43235cc33f: Pushed
04d93f6acee7: Pushing [=====] 175.1kB
ce495f7b0b7d: Pushing [=====] 31.86MB
9c70f446fbe2: Pushing [=====] 7.168kB

```

The screenshot shows the ACR console interface. On the left, there is a sidebar with navigation links: 概览 (Overview), 仓库管理 (Repository Management), 镜像仓库 (Image Repository), 命名空间 (Namespace), 代码源 (Code Source), and 访问凭证 (Access Token). The '镜像仓库' (Image Repository) link is highlighted. The main area displays a table of repositories:

仓库名称	命名空间	仓库状态	仓库类型	仓库地址	创建时间	最近更新时间	操作
fraudapp	fraudapp25	✓ 正常	公开	...	2025-06-23 06:20:06	2025-06-23 06:26:58	管理 删除 置为私有
fraudfront	fraudapp25	✓ 正常	公开	...	2025-06-23 06:31:44	2025-06-23 06:32:57	管理 删除 置为私有

以下文件根据需要修改即可

The screenshot shows a code editor with a dark theme. On the left, there is a file tree showing the project structure:

- > dockers
- > docs
- > fraud-api
- > fraud-biz
- > fraud-commons
- > fraud-repository
- > fraud-starter
- < fraudfront
 - > .idea
 - > .vscode
 - > dist
 - > node_modules
 - > public
 - > src
 - ∅ .gitignore
 - ∅ Dockerfile
 - <> index.html
 - ≡ nginx.conf
- ∅ package.json
- ∅ package-lock.json
- M README.md

The right pane contains the content of the 'nginx.conf' file, which is a configuration for an Nginx server. The line `proxy_pass http://fraud.fraudapp.svc.cluster.local/fraud/;` is highlighted with a red box.

```

# 如果没有upgrade头，则$connection_upgrade$close，否则为upgrade
map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}

server {
    listen 80;
    server_name localhost;
    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
    location /fraud/ {
        proxy_pass http://fraud.fraudapp.svc.cluster.local/fraud/;
        proxy_buffering off;
        proxy_request_buffering off;
    }
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }
}

```

2. 阿里云或者K8S环境部署说明

本内容只是部署测试，生产环境部署需要调整部署策略，如上面的部署应用章节说明。

2.1 部署环境要求

因为涉及到基础以来组件：mysql、redis、rocketmq的部署。最小资源需求是8C16G的两个节点。同时还要对其进行测试验证。k8s版本1.26.x版本为佳

并且镜像仓库中已经包含如下两个镜像内容：

The screenshot shows the AliCloud Container Registry interface. On the left, there's a sidebar with options like '概览', '仓库管理', '镜像仓库' (selected), '命名空间', '代码源', and '访问凭证'. The main area has a search bar at the top with 'fraudapp25' selected. Below it is a table with columns: '仓库名称' (Repository Name), '命名空间' (Namespace), '仓库状态' (Repository Status), '仓库类型' (Repository Type), '仓库地址' (Repository Address), '创建时间' (Created Time), '最近更新时间' (Last Updated Time), and '操作' (Operations). Two entries are listed: 'fraudapp' and 'fraudfront', both in the 'fraudapp25' namespace and in a '正常' (Normal) status.

准备好阿里云的ACK容器服务(本地K8S基本操作流程一致)如下下图：

The screenshot shows the AliCloud Container Service ACK Cluster List page. The left sidebar includes '容器服务 ACK' (Container Service ACK), '集群列表' (Cluster List), '镜像服务' (Image Service), '非模板' (Non-template), and '应用市场' (Application Market). The main area has a search bar with '创建集群' (Create Cluster) and '集群模板' (Cluster Template) buttons. A table lists clusters with columns: '集群名称/ID' (Cluster Name/ID), '标签' (Labels), '集群类型' (Cluster Type), '集群规格' (Cluster Specification), '地域' (Region), '集群状态' (Cluster Status), '节点数' (Node Count), and '操作' (Operations). One cluster, 'cluster-WHXfKI', is highlighted with a red box. It has an ID of 'cacc5bb10874e43ca98fdd9813cd8edcc', is managed by 'ACK 托管集群' (ACK Managed Cluster), and is in '基础版' (Basic Edition) specification. It is located in '华东1 (杭州)' (East China 1 (Hangzhou)), is '运行中' (Running), and has 2 nodes.

2.2 部署步骤与验证

2.2.1 创建名称为**fraudapp**的namespace。

如下图所示。

The screenshot shows the AliCloud Container Service ACK Namespace Management page. The left sidebar includes '集群信息' (Cluster Information), '节点管理' (Node Management), '节点池' (Node Pool), '节点' (Nodes), '组件管理' (Component Management), '命名空间与配额' (Namespace and Quota), '工作负载' (Workload), '无状态' (Stateless), '有状态' (Stateful), and '守护进程集' (DaemonSet). The main area has a search bar with '请输入搜索内容' (Enter search content). Below it is a table titled '命名空间 Namespace' (Namespace). The table has columns: '名称' (Name), '标签' (Labels), '状态' (Status), '创建时间' (Created Time), and '操作' (Operations). A new row is being created, with '名称' (Name) set to 'fraudapp'. Other existing namespaces listed include 'ack-csi-fuse', 'arms-prom', 'default', 'kube-node-lease', 'kube-public', and 'kube-system'. Each namespace entry includes a '资源配额与限制' (Resource Quota and Limit) link and edit and more options buttons.

2.2.2 部署MySQL

使用工程deploy目录下的mysql-deploy.yaml文件即可自动部署。同时会生成对应的Service信息等。确认容器组，服务等资源已经存下切正常。同时创建mysql之后对数据进行初始化，导入工程下面sql目录下面create_schema.sql文件。如下图信息：

The screenshot shows three main sections of the Kubernetes UI:

- Deployment**: A table listing a single deployment named "mysql" in the "fraudapp" namespace. It has 1/1 replicas, was created on 2025-06-23 06:56:59, and last updated on 2025-06-23 06:59:41. An action button "详情 | 编辑 | 伸缩 | 更多" is available.
- Service**: A table listing a single service named "mysql" of type "NodePort". It maps port 3306 to 31306/TCP. The external IP is listed as "None". An action button "更新 | YAML 编辑 | ..." is available.
- Secret**: A table listing a single secret named "mysql-secret" with type "Opaque". It was created on 2025年6月23日 06:56:59. An action button "编辑 | YAML 编辑 | 删除" is available.

Below the Service section, there is a terminal window showing MySQL command-line output:

```
-> UNIQUE KEY `uq_user`(`user`)
-> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci COMMENT='通知人员表
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_fraud |
+-----+
| alert_record
| blacklist
| notify_user
| rule
| transaction
+-----+
5 rows in set (0.00 sec)
```

2.2.3 部署Redis

使用redis-deploy.yaml脚本进行redis服务的创建。确保redis服务正常运行。如下图所示：

<input type="checkbox"/> 名称	命名空间	标签	容器组数量	创建时间	更新时间	操作
<input type="checkbox"/> mysql	fraudapp		1/1	2025-06-23 06:56:59	2025-06-23 06:59:41	详情 编辑 伸缩
<input type="checkbox"/> redis	fraudapp	app:redis	1/1	2025-06-23 07:32:06	2025-06-23 07:35:05	详情 编辑 伸缩

2.2.4 部署RocketMQ以及配套组件

使用rocketmq-deploy.yaml文件进行rocketmq的部署。确保rocketmq可以正常运行以及使用。

<input type="checkbox"/> 使用镜像创建	<input type="checkbox"/> 使用YAML创建资源	命名空间	app:redis	请输入	
<input type="checkbox"/> 名称	命名空间	标签	容器组数量	创建时间	更新时间
<input type="checkbox"/> mysql	fraudapp		1/1	2025-06-23 06:56:59	2025-06-23 06:59:41
<input type="checkbox"/> redis	fraudapp	app:redis	1/1	2025-06-23 07:32:06	2025-06-23 07:35:05
<input type="checkbox"/> rocketmq-broker	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:45:41
<input type="checkbox"/> rocketmq-dashboard	fraudapp	app:rocketmq-dashboard	1/1	2025-06-23 07:37:09	2025-06-23 07:48:13
<input type="checkbox"/> rocketmq-nameserver	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:46:33

2.2.5 部署部署Fraud前后端服务

以上环境准备好之后，可以使用fraud-deploy.yaml创建deployment后端服务。**注意：如果不是使用工程自带的k8s部署环境的话，需要工程下面的application.yaml配置文件，以确保文件正常运行。**

使用fraudfront-deploy.yaml文件部署前端服务。**同样，如果使用不同的环境配置需要修改对应的配置文件，如5章节镜像制作中所示。**

部署fraud后端服务截图如下：

<input type="checkbox"/> 名称	命名空间	标签	容器组数量	创建时间	更新时间
<input type="checkbox"/> fraud	fraudapp	app:fraud	5/5	2025-06-23 07:53:09	2025-06-23 08:05:21
<input type="checkbox"/> mysql	fraudapp		1/1	2025-06-23 06:56:59	2025-06-23 06:59:41
<input type="checkbox"/> redis	fraudapp	app:redis	1/1	2025-06-23 07:32:06	2025-06-23 07:35:01
<input type="checkbox"/> rocketmq-broker	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:45:41
<input type="checkbox"/> rocketmq-dashboard	fraudapp	app:rocketmq-dashboard	1/1	2025-06-23 07:37:09	2025-06-23 07:48:11
<input type="checkbox"/> rocketmq-nameserver	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:46:31

容器组 | 访问方式 | 事件 | 容器伸缩 | 历史版本 | **日志** | 监控 | 成本洞察 | 触发器 |

Pod : fraud-69c74f5df4-bcffj Container : fraud 显示行数 : 100 显示上个容器退出时的日志 刷新 下载

```

{POST [/rule/create]}: createRule(HttpServletRequest,RuleCreateForm)
{GET [/rule/current/rule]}: getCurrentRule(HttpServletRequest)
2025-06-23 08:04:47.248 [main] [DEBUG] Mappings -
    c.f.w.p.r.FraudTXRestController:
        {POST [/tx/evaluate/check]}: TxEvaluate(HttpServletRequest,String,TransactionForm)
2025-06-23 08:04:47.250 [main] [DEBUG] Mappings -
    c.f.w.p.r.TransactionRestController:
        {GET [/transaction/queryByPage]}: queryByPage(HttpServletRequest,TXQueryForm)
        {GET [/transaction/detail]}: detailAlertDO(HttpServletRequest,Long)
        {DELETE [/transaction/delete]}: deleteById(HttpServletRequest,String)
2025-06-23 08:04:47.251 [main] [DEBUG] Mappings -
    c.f.a.f.FraudFeignAPIImpl:
        {POST [/feign/tx/evaluate]}: fraudTxEvaluate(FraudTxEvaReqDTO)
2025-06-23 08:04:47.255 [main] [DEBUG] Mappings -
    o.s.b.a.w.s.e.BasicErrorController:
        { [/error]}: error(HttpServletRequest)
        { [/error], produces [text/html]}: errorHtml(HttpServletRequest,HttpServletResponse)
2025-06-23 08:04:47.459 [main] [DEBUG] Mappings - 'beanNameHandlerMapping' {}
2025-06-23 08:04:49.757 [main] [DEBUG] InternalLoggerFactory - Using SLF4J as the default logging framework
2025-06-23 08:04:52.433 [main] [INFO ] FraudApplication - Started FraudApplication in 23.902 seconds (process running for 25.462)

```

hpa自动扩容配置已经测试生效：

指标伸缩 (HPA) | 创建 | 指标伸缩对象最多一个 | HPA 和 CronHPA 兼容规则

名称	目标使用率	当前使用率	最小副本数	最大副本数	当前副本数	创建时间	操作
fraud-hpa	CPU : 30%		2	5	2	2025-06-23 07:53:10	查看YAML 状态 事件 编辑 删除

工作台 账号全部资源 华东1(杭州) 搜索... 费用 ICP 备案 企业 支持 工单 99

状态 就绪: 2/5个, 已更新: 5个, 可用: 2个 展开现状详情 ▾

容器组	访问方式	事件	容器伸缩	历史版本	日志	监控	成本洞察	触发器	操作	
名称	镜像	状态 (全部) ▾			监控	重启次数	Pod IP	节点	创建时间	操作
fraud-69c74f5df4-bcffj	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2	Running	0	172.16.0.84	cn-hangzhou.192.16 8.56.50	2025-06-23 08:04:26	192.168.56.50	详细		
fraud-69c74f5df4-bf5kv	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2	Running	0	172.16.0.83	cn-hangzhou.192.16 8.56.50	2025-06-23 08:04:26	192.168.56.50	详细		
fraud-69c74f5df4-j855l	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2	Running	0	172.16.0.21	cn-hangzhou.192.16 8.56.51	2025-06-23 08:04:41	192.168.56.51	详细		
fraud-69c74f5df4-nrzwv	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2	Running	0	172.16.0.20	cn-hangzhou.192.16 8.56.51	2025-06-23 07:53:10	192.168.56.51	详细		
fraud-69c74f5df4-pg4sz	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25/fraudapp:v2	Running	0	172.16.0.82	cn-hangzhou.192.16 8.56.50	2025-06-23 07:53:10	192.168.56.50	详细		

前端部署可以使用fraudfront-deploy.yaml部署进行查看。

2.3 测试部署

成功创建后完整的服务如下：

使用镜像创建 使用YAML创建资源 命名空间 fraudapp 请输入

<input type="checkbox"/> 名称	命名空间	标签	容器组数量	创建时间	更新时间
<input type="checkbox"/> fraud	fraudapp	app:fraud	2/2	2025-06-23 07:53:09	2025-06-23 08:05:
<input type="checkbox"/> fraudfront-deployment	fraudapp	app:fraudfront	2/2	2025-06-23 09:08:21	2025-06-23 09:08:
<input type="checkbox"/> mysql	fraudapp		1/1	2025-06-23 06:56:59	2025-06-23 06:59:
<input type="checkbox"/> redis	fraudapp	app:redis	1/1	2025-06-23 07:32:06	2025-06-23 07:35:
<input type="checkbox"/> rocketmq-broker	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:45:
<input type="checkbox"/> rocketmq-dashboard	fraudapp	app:rocketmq-dashboard	1/1	2025-06-23 07:37:09	2025-06-23 07:48:
<input type="checkbox"/> rocketmq-nameserver	fraudapp		1/1	2025-06-23 07:37:09	2025-06-23 07:46:

测试滚动更新：至少保留一个pod服务。

不可用Pod最大数量:1

注解
deployment.kubernetes.io/revision:2
标签
app:fraud

状态 就绪: 1/2个, 已更新: 2个, 可用: 1个 展开现状详情 ▾

容器组	访问方式	事件	容器伸缩	历史版本	日志	监控	成本洞察	触发器		
名称	镜像	状态 (全部) ▾			监控	重启次数	Pod IP	节点	创建时间	操作
fraud-69c74f5df4-j85l	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25 /fraudapp:v2	Running			172.16.0 .21	cn-hangzhou.192.1 68.56.51 192.168.56.51	2025-06-23 08:04:41	详情 YAML 编辑 终端 更多 ▾		
fraud-69c74f5df4-nrzv	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25 /fraudapp:v2	Terminating			172.16.0 .20	cn-hangzhou.192.1 68.56.51 192.168.56.51	2025-06-23 07:53:10	详情 YAML 编辑 终端 更多 ▾		
fraud-7d9656688f-bs7wm	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25 /fraudapp:release	ContainerCreating ⓘ			172.16.0 .21	cn-hangzhou.192.1 68.56.50 192.168.56.50	2025-06-23 09:33:34	详情 YAML 编辑 终端 更多 ▾		
fraud-7d9656688f-h5w59	crpi-1nz5mr3oba11j6z.cn-hangzhou.personal.cr.aliyuncs.com/fraudapp25 /fraudapp:release	ContainerCreating ⓘ			172.16.0 .20	cn-hangzhou.192.1 68.56.50 192.168.56.50	2025-06-23 09:33:34	详情 YAML 编辑 终端 更多 ▾		

七. 功能测试与验证

1. 测试环境业务验证

1.1 目前已经部署到阿里云测试环境(<http://115.29.243.179:8090/>)

1.2 可以检查规则，黑名单，通知查看等交易检查等方面进行测试。

1.2.1 交易规则查看与更新

反欺诈检测

- [检测规则设置](#)
- [黑名单管理](#)
- [通知人员管理](#)
- [通知查询](#)
- [交易记录](#)
- [反欺诈测试](#)

检测规则设置

欺诈交易金额:

欺诈交易时段 (0-24格式):

是否开启黑名单:

1.2.2 黑名单控制

反欺诈检测

[检测规则设置](#)

[黑名单管理](#)

[通知人员管理](#)

[通知查询](#)

[交易记录](#)

[反欺诈测试](#)

黑名单管理

查询
新增黑名单

账号	姓名	原因	操作
sfpt12	zhangs23	欺诈开始了	详情 删除
sds12234	张三	张三黑名单 测试 验证	详情 删除
zhags1n1322	zsdha12g1	异常交易1信息港1	详情 删除
zhags1n13	zsdha1g1	异常交易1信息港1	详情 删除

上一页
第 1 页, 共 1 页
下一页

1.2.3 检查交易是否存在欺诈行为

反欺诈检测

[检测规则设置](#)

[黑名单管理](#)

[通知人员管理](#)

[通知查询](#)

[交易记录](#)

[反欺诈测试](#)

账号:

金额:

时间戳 (毫秒):

描述:

提交测试 检测结果已经原因

测试结果

是否欺诈: 是

检测说明:

检测交易信息:

```
{"transactionId":"23432dssqf12323","account":"34r23sd","amount":"334 Transaction amount exceeds: 70000.0}
```

2. 关键检测接口集成测试验证(post, /**fraud/tx/evaluate**)

2.1 TXEvalEngine 并发测试验证主要测试目标场景如下。

- 正常执行流程
- 锁竞争失败抛出异常
- 并发环境下仅一个线程成功执行业务逻辑的关键验证

测试方法	说明
testEvaluateTx_NormalFlow	正常命中所有规则，触发通知。
testEvaluateTx_ConcurrentLockPrevented	模拟锁获取失败，触发重复提交异常。
testConcurrentLocking_OnlyOneThreadShouldPass	并发模拟下，确保只有一个线程执行业务，其余被锁拦截。

代码块

```
1  /**
2   * 测试方法说明:
3   * | 测试方法 | 说明
4   * | ----- | -----
5   * | `testEvaluateTx_NormalFlow` | 正常命中所有规则, 触发通知。
6   * | `testEvaluateTx_ConcurrentLockPrevented` | 模拟锁获取失败, 触发重复提交异常。
7   * | `testConcurrentLocking_OnlyOneThreadShouldPass` | 并发模拟下, 确保只有一个线程执行业务, 其余被锁拦截。 |
8   */
9
10 /**
11  * 并发控制单元测试
12 */
13 @ExtendWith(MockitoExtension.class)
14 public class TXEvalEngineLockTest {
15
16     private TXEvalEngine txEvalEngine;
17     private BlackListService blackListService;
18     private TxRuleLoader txRuleLoader;
19     private TransactionService transactionService;
20     private NotifyService notifyService;
21     private LockService lockService;
22
23     @BeforeEach
24     void setUp() {
25         blackListService = mock(BlackListService.class);
26         txRuleLoader = mock(TxRuleLoader.class);
27         transactionService = mock(TransactionService.class);
28         notifyService = mock(NotifyService.class);
29         lockService = mock(LockService.class);
30
31         txEvalEngine = new TXEvalEngine(blackListService, txRuleLoader,
32                                         transactionService, notifyService, lockService);
33     }
34 }
```

```
33     // 默认规则设置
34     RuleDO amountRule = new RuleDO(); amountRule.setValue("1000");
35     RuleDO blackRule = new RuleDO(); blackRule.setValue("true");
36     RuleDO timeRule = new RuleDO(); timeRule.setValue("0 - 23");
37
38     lenient().when(txRuleLoader.loadRule("amount")).thenReturn(amountRule);
39
40     lenient().when(txRuleLoader.loadRule("blacklist")).thenReturn(blackRule);
41
42     lenient().when(txRuleLoader.loadRule("off_hours")).thenReturn(timeRule);
43 }
44
45 @Test
46 void testEvaluateTx_NormalFlow() {
47     TransactionDO tx = buildTx("T123", "acct1", "2000");
48
49     when(blackListService.isBlackListExist("acct1")).thenReturn(true);
50     when(lockService.tryLock(eq("T123"))).thenReturn(true);
51     doNothing().when(lockService).unlock("T123");
52
53     List<String> result = txEvalEngine.evaluateTx(tx);
54
55     assertFalse(result.isEmpty());
56     verify(transactionService).createTransaction(tx);
57     verify(notifyService).sendTransactionCheckNotify(eq(tx), anyList());
58     verify(lockService).unlock("T123");
59 }
60
61 @Test
62 void testEvaluateTx_ConcurrentLockPrevented() {
63     TransactionDO tx = buildTx("T999", "acctX", "2000");
64
65     when(lockService.tryLock(eq("T999"))).thenReturn(false);
66
67     RuntimeException ex = assertThrows(RuntimeException.class, () -> {
68         txEvalEngine.evaluateTx(tx);
69     });
70
71     assertTrue(ex.getMessage().contains("重复请求") || ex instanceof
72 com.sct.bizmsg.exception.BusinessException);
73 }
74
75 @Test
76 void testConcurrentLocking_OnlyOneThreadShouldPass() throws
77 InterruptedException {
78     TransactionDO tx = buildTx("T888", "acctY", "999");
79 }
```

```

75
76     AtomicInteger attemptCount = new AtomicInteger(0);
77     when(lockService.tryLock(eq("T888"))).thenAnswer(invocation ->
78         attemptCount.incrementAndGet() == 1);
79         doNothing().when(lockService).unlock("T888");
80
81     int threadCount = 5;
82     ExecutorService executor =
83     Executors.newFixedThreadPool(threadCount);
84     CountDownLatch latch = new CountDownLatch(threadCount);
85
86     for (int i = 0; i < threadCount; i++) {
87         executor.submit(() -> {
88             try {
89                 txEvalEngine.evaluateTx(tx);
90             } catch (Exception ignored) {
91             } finally {
92                 latch.countDown();
93             }
94         });
95     }
96
97     latch.await();
98     executor.shutdownNow();
99
100    verify(lockService, times(threadCount)).tryLock(eq("T888"));
101    verify(lockService, times(1)).unlock("T888");
102    verify(transactionService, times(1)).createTransaction(any());
103
104    private TransactionDO buildTx(String id, String account, String amount)
105    {
106        TransactionDO tx = new TransactionDO();
107        tx.setTransactionId(id);
108        tx.setAccount(account);
109        tx.setAmount(amount);
110        tx.setTransactionTime(new Date());
111        tx.setDescription("测试交易");
112        return tx;
113    }
114 }

```

2.2 Mock mvc接口调用测试模拟测试。

主要验证返回数据格式，字段等内容。

```
1 import com.fasterxml.jackson.databind.ObjectMapper;
2 import com.fraud.FraudApplication;
3 import com.fraud.biz.txengine.TXEvalEngine;
4 import com.fraud.web.platform.form.TransactionForm;
5 import org.junit.jupiter.api.Test;
6 import org.junit.jupiter.api.extension.ExtendWith;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import
9     org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc
10    ;
11 import org.springframework.boot.test.context.SpringBootTest;
12 import org.springframework.boot.test.mock.mockito.MockBean;
13 import org.springframework.http.MediaType;
14 import org.springframework.test.context.junit.jupiter.SpringExtension;
15 import org.springframework.test.web.servlet.MockMvc;
16 import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
17 import org.springframework.test.web.servlet.result.MockMvcResultMatchers;
18
19 import java.util.Date;
20 import java.util.List;
21
22 import static org.mockito.ArgumentMatchers.any;
23 import static org.mockito.Mockito.when;
24 import static
25     org.springframework.test.web.client.match.MockRestRequestMatchers.jsonPath;
26 import static
27     org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
28 import static
29     org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
30
31
32 @ExtendWith(SpringExtension.class)
33 @SpringBootTest(classes = FraudApplication.class)
34 @AutoConfigureMockMvc
35 class FraudTXRestController {
36
37     @Autowired
38     private MockMvc mockMvc;
39
40     @MockBean
41     private TXEvalEngine txEvalEngine;
42
43     @Test
44     void testCheckTxFraudAPI() throws Exception {
45         TransactionForm form = new TransactionForm();
```

```

43         form.setTransactionId("tx123");
44         form.setAccount("acc001");
45         form.setAmount("500000");
46         form.setTransactionTime(new Date());
47
48     List<String> fakeReasons = List.of("Amount exceeds limit");
49     when(txEvalEngine.evaluateTx(any())).thenReturn(fakeReasons);
50
51     mockMvc.perform(post("/tx/evaluate/check")
52                     .contentType(MediaType.APPLICATION_JSON)
53                     .content(new
54             ObjectMapper().writeValueAsString(form)))
55                     .andExpect(status().isOk())
56
57     .andExpect(MockMvcResultMatchers.jsonPath("$.data.checkResult").value(true));
58 }
59
60 @Test
61 void testCheckTx_Success() throws Exception {
62     String json = """
63         "transactionId": "22864539stds",
64         "account": "ew1474",
65         "amount": "45659666",
66         "transactionTime": 1750477432000,
67         "description": "交易信454554息"
68     """
69     """
70
71     mockMvc.perform(post("/tx/evaluate/check").contentType(MediaType.APPLICATION_JSON).content(json))
72                     .andExpect(status().isOk())
73
74     .andExpect(MockMvcResultMatchers.jsonPath("$.code").value("SUCCESS"))
75     .andExpect(
76         MockMvcResultMatchers.jsonPath("$.data.checkResult").isBoolean());
77 }

```

2.3 部署验证与恢复测试

2.3.1 自动扩容测试

自动扩容使用时k8s自带的HPA机制。设定监控metrics为内存用量30%则自动扩容范围2-10个副本信息。hpa文件如下：

代码块

```
1  apiVersion: autoscaling/v2
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: fraud-hpa
5    namespace: ma0uqkj4-fraud
6  spec:
7    scaleTargetRef:
8      apiVersion: apps/v1
9      kind: Deployment
10     name: fraud
11   minReplicas: 2
12   maxReplicas: 10
13   metrics:
14     - type: Resource
15       resource:
16         name: cpu
17       target:
18         type: Utilization
19         averageUtilization: 30
```

2.3.2 节点故障测试

操作流程如下

- a. 在一个节点上部署一个简单的应用，并设置两个副本。
- b. 使用 `kubectl drain <节点名称>` 模拟节点宕机。
- c. 观察Pod迁移到其他节点，并恢复正常运行。

当集群从两个节点变成一个节点后，应用仍然正常运行：

The screenshot shows the ACK Cluster Management interface. The top navigation bar includes '容器服务 ACK' (Container Service ACK), '集群列表' (Cluster List), '集群: cluster-WHXfKI' (Cluster: cluster-WHXfKI), and '帮助' (Help). The main title is '节点' (Nodes). A yellow warning banner at the top right says: '如您需转按量付费节点为包年包月, 建议您切勿勾选“转为包年包月磁盘”。因为若勾选“转为包年包月磁盘”后会将按量计费云盘全部转为包年包月, 导致存储挂载的云盘不能进行卸载操作。' Below the banner are search and filter fields: '名称' (Name), '请输入搜索内容' (Enter search content), '所有节点池' (All Node Pools), '标签检索' (Label Search), and '只显示不可调度节点' (Only show non-schedulable nodes). A table lists nodes:

名称/IP 地址/实例 ID	标签	所属节点池	角色/状态	配置	容器组 (已分配量/总额度)	操作
cn-hangzhou.192.168.56.51 192.168.56.51 i-bp17ftq8kiqnbdd9bbh		default-nodepool	Worker ⓘ 就绪	抢占式实例 ecs.c6.2xlarge 8 vCPU 16 GiB 可用区	27/64	详情 监控 更多 ▾

无状态 Deployment

使用镜像创建	使用YAML创建资源	命名空间 fraudapp	请输入	操作
<input type="checkbox"/> fraud	fraudapp	app:fraud	2/2	2025-06-23 12:04:19 2025-06-23 12:08:19 详情 编辑 伸缩
<input type="checkbox"/> fraudfront-deployment	fraudapp	app:fraudfront	2/2	2025-06-23 09:36:34 2025-06-23 09:36:45 详情 编辑 伸缩

2.3.3 优雅升级与切换

在deployment文件里面已经配置相应的滚动更新策略和存活探针检测，结合k8s的自动调度功能可以观察到对应结果。如下图

```
71      initialDelaySeconds: 10
72      periodSeconds: 10
73      livenessProbe:
74        httpGet:
75          path: /fraud/actuator/health
76          port: 8080
77          initialDelaySeconds: 20
78          periodSeconds: 10
79      progressDeadlineSeconds: 600
80      revisionHistoryLimit: 15
81      minReadySeconds: 10
82      strategy:
83        type: RollingUpdate      # 滚动更新策略
84        rollingUpdate:
85          maxSurge: 2            # 升级过程中最多可以比原先设置的副本数多出的数量
86          maxUnavailable: 1      # 升级过程中最多有多少个POD处于无法提供服务的状态
```

3. 单元测试

- 单元测试放在starter模块test目录下面。主要的业务逻辑都覆盖了。13个测试类，共计58个测试用例

Overall Coverage Summary

Package	Class, %	Method, %	Branch, %	Line, %
all classes	58.5% (31/53)	22.6% (107/474)	5.6% (17/301)	20.5% (231/1128)

Coverage Breakdown

Package	Class, %	Method, %	Branch, %	Line, %
com.fraud	100% (1/1)	100% (2/2)		100% (2/2)
com.fraud.aop	100% (1/1)	100% (15/15)		100% (37/37)
com.fraud.api.feign	100% (1/1)	50% (1/2)	0% (0/12)	7.4% (2/27)
com.fraud.api.request.fraud	0% (0/1)	0% (0/11)		0% (0/11)
com.fraud.biz.cache.redis	100% (2/2)	25.9% (21/81)	20% (2/10)	44.8% (64/143)
com.fraud.biz.enums	0% (0/1)	0% (0/3)		0% (0/6)
com.fraud.biz.notify	50% (1/2)	4.3% (1/23)	0% (0/5)	6.2% (2/32)
com.fraud.biz.notify.impl	100% (1/1)	50% (1/2)	0% (0/4)	8.3% (2/24)
com.fraud.biz.notify.rocketmq.impl	100% (1/1)	50% (1/2)	0% (0/2)	22.2% (2/9)
com.fraud.biz.notify.sender	100% (3/3)	50% (3/6)		50% (6/12)
com.fraud.biz.service.impl	85.7% (6/7)	20.5% (9/44)	7.1% (4/56)	13.2% (30/227)
com.fraud.biz.txengine	100% (2/2)	28.6% (2/7)	0% (0/32)	15.2% (10/66)
com.fraud.commoncons.cons	0% (0/1)	0% (0/1)		0% (0/1)
com.fraud.commoncons.enums	0% (0/2)	0% (0/15)	0% (0/4)	0% (0/42)
com.fraud.configuration.datasource	100% (1/1)	100% (5/5)		100% (8/8)
com.fraud.configuration.mvc	100% (2/2)	87.5% (7/8)		74.1% (20/27)
com.fraud.consumer	100% (1/1)	50% (1/2)	0% (0/2)	11.1% (2/18)
com.fraud.repository.app.model	40% (2/5)	28.8% (32/111)	17.2% (11/64)	28.8% (32/111)
com.fraud.web.platform.form	0% (0/10)	0% (0/96)		0% (0/96)
com.fraud.web.platform.rest	100% (6/6)	23.1% (6/26)	0% (0/110)	5.5% (12/217)
com.fraud.web.platform.vo	0% (0/2)	0% (0/12)		0% (0/12)

2. 主要交易验证接口性能测试

八. 程序中后续扩展目标

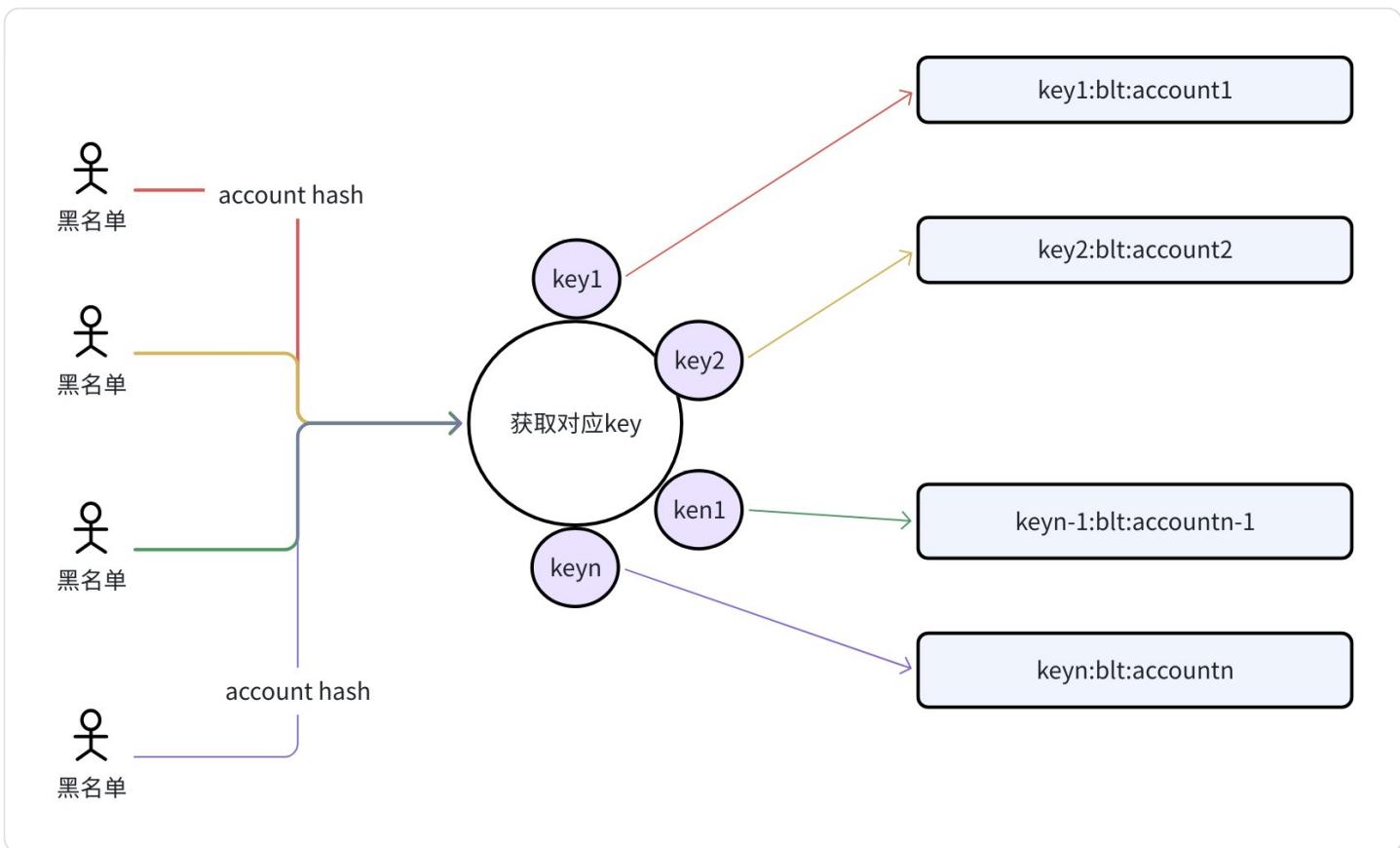
1. nacos的注册配置中心集成

集成nacos方便服务调用以及将配置文件放到配置中心等。

2. 针对BalckList缓存大Key的问题

一个key会是的数据变得很大。我们可以采用分散key的方式。

利用一致性hash方法，建立一个hash映射关系，按照黑名单的account信息进行hash之后，从而可以分散key在缓存中位置，减少big key的风险。



3. 高并发场景下可用性保障，如熔断/限流等