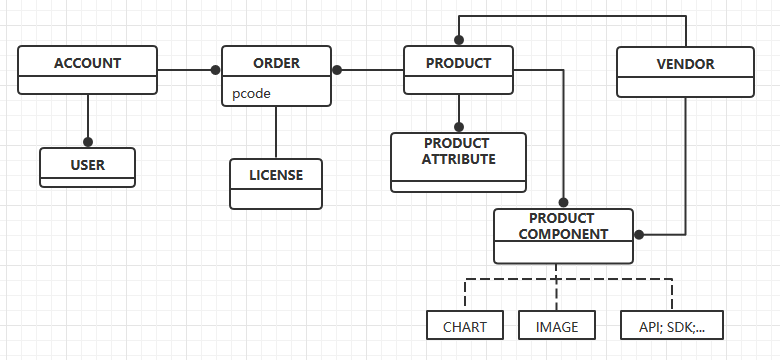
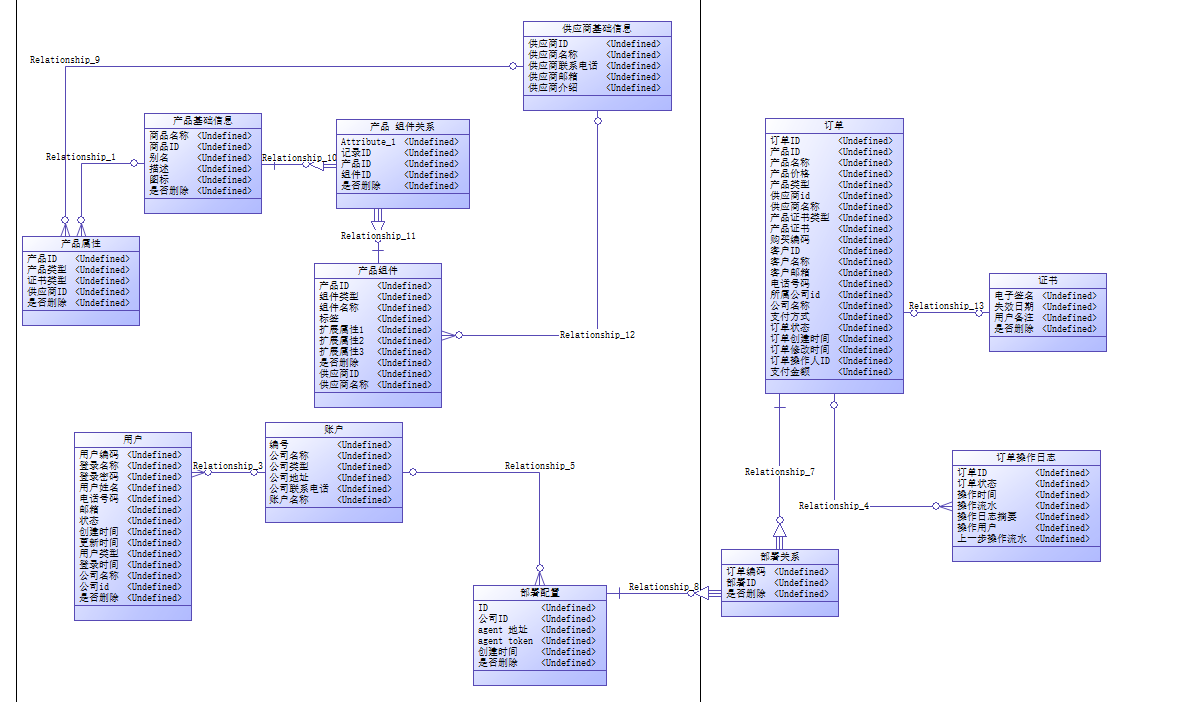
# 模型设计

## 逻辑模型



1. Account 为一个账号 包含多个用户信息。
2. Vendor 是供应商，供应商拥有多个产品以及多个产品组件。
3. Product 由一个或多个产品组件 (Product component) 构成
4. Product component是image、chart、API 、或SDK等文件形式Artifact
5. 客户选中一个产品下单购买，一个Order购买一个产品
6. Vendor License API 验证PCODE（由线下生成）给order授权一个license

## 物理模型



# 服务与API设计

## 服务列表

|  |  |
| --- | --- |
| **服务名** | **功能描述** |
| license service | 供应商grant License的服务 |
| product service | 商品服务 |
| account service | 账户服务 |
| order service | 订单相关服务 |
| Vendor service | 供应商管理的相关服务 |
| market service | 市场后台服务 |
| docker auth | 仓库认证服务 |
| registry | 镜像仓库Registry原生API 提供 image 下载服务 |
| chart museum | Chart文件仓库 |

## API

### market service API

|  |  |  |
| --- | --- | --- |
| **API Name** | **描述** | **操作** |
| purchase | 下单购买 | POST |
| getRepToken | 获取仓库访问token | GET |
| grantRepPermission | 账号镜像赋权 | POST |
| getChart | 获取chart文件 | GET |
| getLicense | 获取license | GET |

#### API Schema

…..

### Vendor License API

|  |  |  |
| --- | --- | --- |
| **接口名** | **描述** | **操作** |
| 证书生成接口 | 根据purchase code 生成 license | POST |

#### License Schema

……

### product server API

|  |  |  |  |
| --- | --- | --- | --- |
|  | **接口名** | **描述** | **操作** |
| 商品API | createProduct | 创建商品 | POST |
| queryProduct | 查询商品列表 | GET |
| getProduct | 查询商品 | GET |
| updateProduct | 修改商品 | PUT |
| deleteProduct | 删除商品 | DELETE |
| 商品组件API | createComponent | 创建组件 | POST |
| queryComponent | 查询组件列表 | GET |
| getComponent | 查询组件 | GET |
| updateComponent | 更新组件 | PUT |
| deleteComponent | 删除组件 | DELETE |

### account server API

|  |  |  |  |
| --- | --- | --- | --- |
|  | **API Name** | **描述** | **操作** |
| 账户管理 | createAccount | 创建一个账号 | POST |
| queryAccounts | 查询账号列表 | GET |
| getAccount | 查询账号信息 | GET |
| updateAccount | 更新账号属性 | PUT |
| deleteAccount | 删除账号 | DELETE |
| 用户管理(用户这部分可以考虑用ug实现) | createUser | 创建一个用户 | POST |
| queryUsers | 查询用户列表 | GET |
| getUser | 获取用户信息 | GET |
| updateUser | 更新用户 | PUT |
| deleteUser | 删除用户 | DELETE |
| agent 管理 | createAgent | 创建agent配置 | POST |
| queryAgents | 查询代理列表 | GET |
| getAgent | 查询代理信息 | GET |
| updateAgent | 修改agent | PUT |
| deleteAgent | 删除agent | DELETE |

### Order Server API

|  |  |  |
| --- | --- | --- |
| **API Name** | **描述** | **操作** |
| createOrder | 创建订单 | POST |
| queryOrders | 订单列表查询 | GET |
| getOrder | 查询单个订单信息 | GET |
| updateOrder | 修改订单 | PUT |

### Vendor service API

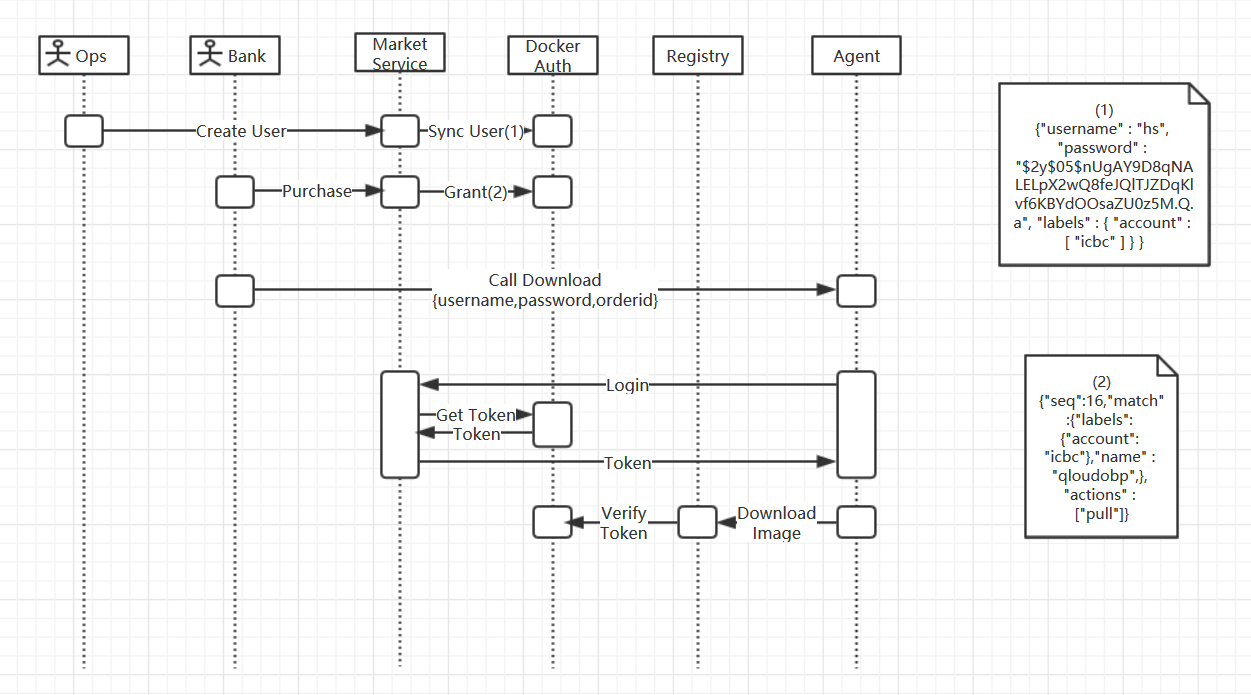
|  |  |  |
| --- | --- | --- |
| **API Name** | **描述** | **操作** |
| createVendor | 创建一个供应商 | POST |
| queryVendors | 供应商列表查询 | GET |
| getVendor | 查询单个供应商信息 | GET |
| updateVendor | 修改供应商 | PUT |

### Registry Service (原生的registry的API)

|  |  |  |
| --- | --- | --- |
| **API Name** | **描述** | **操作** |
| getManifest | 查询镜像文件的manifest | POST |
| getLayer | 获取镜像层文件 | POST |

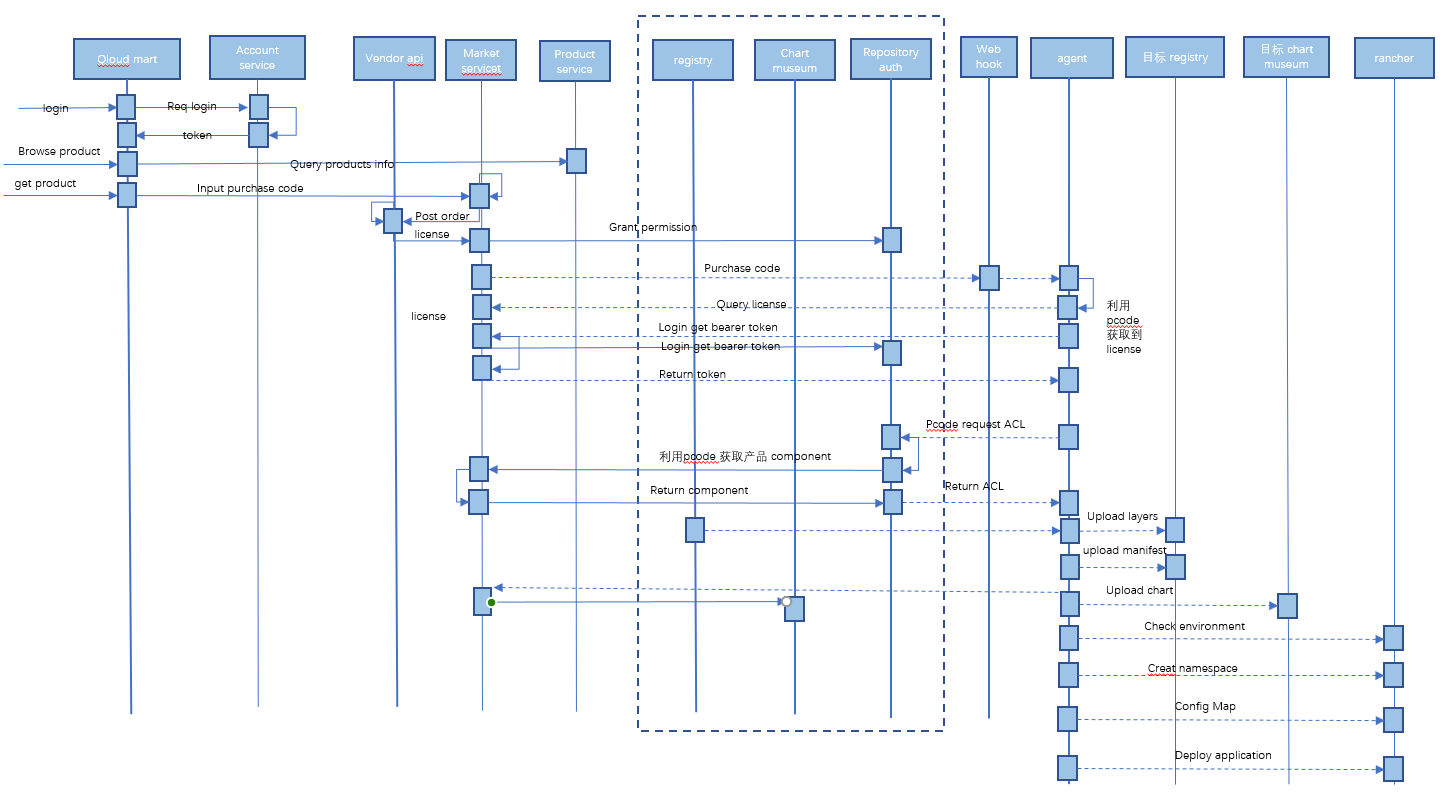
# 安全与认证

## 镜像仓库下载安全与认证流程



1. 用户通过market service 创建registry 账号。
2. 当用户购买商品后market service给该用户所属的？账号赋予相应镜像的访问权限
3. 当银行部署购买的应用时 通知agent去获取购买的商品(通过 order ID 账号 密码)
4. AGENT请求market service 认证接口，market service 再去请求docker auth获取TOKEN
5. 返回token给agent
6. Agent 拿到token 去registry请求，如果镜像访问权限再第二步已经赋予则可成功获取镜像。

# 购买下载部署全流程



1. 用户在qloudmart通过账号密码在account service上进行认证登录。
2. 登录后前端调用product service 获取到商品列表。
3. 用户点击商品购买后输入purchase code 进行下单购买。
4. Marketservice得到purchase code后调用vendor api获取到license在market service端保存。
5. Market service 得到license后 再根据订单的下单账号，以及订单商品的组件信息。再docker auth 中对相应账号进行赋权，以便后面操作image .
6. 发送purchase code和order ID 通知market agent 去获取订单。
7. Agent 获取到通知后 首先请求market service 获取 license.
8. 然后根据订单ID去获取 docker login token.
9. Market service获取到订单ID根据订单的下单账号 到docker auth去登录认证。并从docker auth中获取到token
10. 返回token 给market agent,由于再第5步已经对相应的镜像进行了赋权 ，所以 agent 可以拿着token 直接去registry 下载镜像文件。
11. 通过market service 到chart museum 下载chart文件
12. 上传chart文件到用户环境下的chart museum
13. 以上完成后请求rancher 接口获取相关环境信息
14. 创建rancher namespace
15. 将第7步获取到的license 形成文件再通过config map 同步到用户环境配置中。
16. 调用部署接口进行部署。