1. 项目概况

业务上，通过组合 `UniswapV2` 和 `Compound` 实现永续合约功能

技术上，将前面所学内容串联起来，形成一个完整互通的 DApp

项目共包括 4 部分：

- **core**：最核心的智能合约
- **subgraph**：数据索引服务
- **keeper**：Keeper 服务
- **interface**：前端 UI

2. 业务模型

永续合约

用保证金做抵押，可放大杠杆的类期货交易产品

- 币本位合约：用标的资产作为保证金
- U 本位合约：用 U 作为保证金

核心功能

为了简化，只做固定的一个交易对：`ETH/USDC`

核心功能包括：

- **注册**：注册交易账号
- **充值**：充值到合约账户充当保证金
  - **充值 ETH**：用于做多时
  - **充值 USDC**：用于做空时
- **提现**：从合约账户提取到用户钱包
  - **提现 ETH**：将存款的 ETH 提取到钱包
  - **提现 USDC**：将存款的 USDC 提取到钱包
- **开仓**：通过 FlashSwap 和存借款完成开仓
  - **开多**：存 ETH 借 USDC 实现开多
  - **开空**：存 USDC 借 ETH 实现开空
- **平仓**：通过还款实现平仓
  - **平多**：还 USDC 实现平多
  - **平空**：还 ETH 实现平空
- **限价单**：当预言机价格到达设置的价格时触发开仓
  - **限价开多**：当 ETH 价格回落到指定价格时触发开多
  - **限价开空**：当 ETH 价格上涨到指定价格时触发开空

3. 合约实现

核心合约有两个：

- **AccountRegistry**
- **TradingAccount**

AccountRegistry

主要提供给用户创建交易账户功能

用户调用 `createAccount` 函数，会创建出用户自己为 owner 的**交易账户合约**，即 `TradingAccount` 合约

采用 **Clone** 的方式创建 `TradingAccount` 合约，相比用 **new** 的方式可以省 gas

TradingAccount

用户所有操作都通过该合约，合约主要提供的函数包括：

- **depositETH**: 充值 ETH，同时会存款到 Compound
- **depositUSDC**: 充值 USDC，同时会存款到 Compound
- **withdrawETH**: 提现 ETH，会从 Compound 中赎回存款并提到用户地址
- **withdrawUSDC**: 提现 USDC，会从 Compound 中赎回存款并提到用户地址
- **openLong**: 开多，即看涨 ETH
- **openShort**: 开空，即看跌 ETH
- **closeLong**: 平多
- **closeShort**: 平空
- **limitOpenLong**: 限价单开多
- **limitOpenShort**: 限价单开空
- **cancelLimitOrder**: 取消限价单
- **executeLimitOrder**: 执行限价单，由指定的 keeper 触发

开多

0. 开多之前需要先完成了充值 ETH
1. 调 Uniswap 进行 Swap 操作，用 USDC 兑换出 ETH
2. 回调到 uniswapV2Call 函数，该函数会先拿到 WETH
3. 将 WETH 换成 ETH，然后存款到 Compound，提高抵押资产价值
4. 从 Compound 借出 USDC，并转回给到 Uniswap

开多完成后，交易账户会在 Compound 存款了 ETH，并借款了 USDC

存款的 ETH 有两部分来源，一是用户自己充值的，二是从 Uniswap 通过 FlashSwap 兑换回来的

开空

0. 开空之前需要先完成了充值 USDC
1. 调 Uniswap 进行 Swap 操作，用 WETH 兑换出 USDC
2. 回调到 uniswapV2Call 函数，该函数会先拿到 USDC
3. 将 USDC 存款到 Compound，提高抵押资产价值
4. 从 Compound 借出 ETH，转成 WETH 后转回给到 Uniswap

开空完成后，交易账户会在 Compound 存款了 USDC，并借款了 ETH

存款的 USDC 有两部分来源，一是用户自己充值的，二是从 Uniswap 通过 FlashSwap 兑换回来的

平多

平多时，需要用存款里的 ETH 兑换成 USDC 并还款

1. 根据要还款的 USDC 金额，计算出需要花费多少 ETH
2. 通过 Uniswap FlashSwap 功能先得到 USDC
3. 将 USDC 还款给 Compound
4. 从 Compound 中赎回部分 ETH
5. 将 ETH 支付给到 Uniswap

平多完成后，交易账户在 Compound 里的 ETH 存款会减少，USDC 借款也会减少

平空

平空时，需要用存款里的 USDC 兑换成 ETH 并还款

1. 根据要还款的 ETH 金额，计算出需要花费多少 USDC
2. 通过 Uniswap FlashSwap 功能先得到 ETH
3. 将 ETH 还款给 Compound
4. 从 Compound 中赎回部分 USDC
5. 将 USDC 支付给到 Uniswap

平空完成后，交易账户在 Compound 里的 USDC 存款会减少，ETH 借款也会减少

限价单

限价单不会立即成交，需等到预言机价格超过 `limitPrice` 之后才可执行成交

限价单无法在链上自动成交，需要指定 `keeper` 进行链下的价格监听，当价格满足条件后由 `keeper` 触发执行合约函数

4. Subgraph

核心功能包括：

- 监听每个用户的交易账号的创建
- 监听所有充值、提现、开仓、平仓、限价单等事件
- 更新每个 Position 信息
- 保存每笔交易记录
- 保存和更新每笔 LimitOrder

subgraph.yaml

定义一个 dataSource，监听 `AccountRegistry` 合约的 `AccountCreated` 事件
定义一个 template，监听 `TradingAccount` 合约的各种操作事件

schema.graphql

定义了几种 entity：

- `Position`: 当前持仓
- `LimitOrder`: 限价单
- `DepositLog`: 充值记录
- `WithdrawLog`: 提现记录
- `OpenPositionLog`: 开仓记录
- `ClosePositionLog`: 平仓记录

mappings

- `account-registry.ts`
- `trading-account.ts`

5. Keeper

核心功能包括：

- 实时获取最新的预言机价格
- 从 Subgraph 读取出所有满足触发条件的 LimitOrder
- 触发 executeLimitOrder

代码包结构

- `main.go`: 程序入口
- `conf`: 配置文件
- `contract`: 合约交互
- `subgraph`: 查询 Subgraph
- `cron`: 定时任务
- `executor`: 执行限价单

6. 前端 UI

核心功能主要包括：

- 注册
- 充值
- 开仓
- 查看持仓
- 关仓
- 提现