

第 5 章



超级账本开源项目

本章学习目标

- 了解超级账本(HyperLedger)项目与特点。
- 了解超级账本子项目及开发工具。
- 部署并使用超级账本项目官方提供的超级账本浏览器(HyperLedger Explorer)。

5.1 超级账本项目简介

2015 年 12 月,为了推动区块链技术跨行业应用,Linux 基金会联合金融、制造和科技等行业企业成立了超级账本项目,该开源项目实现了企业级区块链分布式账本底层技术,首次将区块链技术引入商业联盟场景,通过权限控制和安全保障,能够帮助企业、集团、行业或组织部署联盟链平台以支持他们私有业务流程和事务的开展。

为了能够适应不同场景对效率、安全和可信的需求,超级账本项目具有如下特征。

(1) 模块化设计。

采用模块化、可扩展的框架。便于开发人员针对不同需求创建和测试组件,特定的算法可以作为可插拔组件嵌入项目,提供集成解决方案,方便项目的开发。

(2) 易操作使用。

提供丰富且易于使用的 API。这些 API 支持丰富的开发者生态系统的增长,并帮助区块链和分布式账本技术在广泛的行业和用例中扩散。

(3) 可维护管理。

包含可视化监控管理,方便可视化监控网络中各节点的实时运行。

(4) 安全和保密。

项目中的算法、协议和密码技术都由安全专家以及更广泛的开源社区定期审查和

审核。

超级账本社区目前拥有 300 家左右全球知名企业和机构会员,共有 14 个子项目在运行维护,提出和实现了许多创新设计和理念,包括权限、审计、多通道、细粒度隐私保护、背书—共识—提交模型,以及可拔插、可扩展的实现框架,这些都对区块链技术和产业的发展产生了深远的影响。

1. 重要的子项目简介

1) Fabric 项目

Fabric 是当前最流行的超级账本框架,2015 年 12 月由 IBM、DAH 等企业联合贡献到社区。该项目采用模块化架构,采用 Golang 或 JavaScript 语言编写智能合约(也称链码,Chaincode),并在 Docker 容器中运行,通过可插拔组件提供共识和会员服务等功能。Fabric 项目可以托管和执行智能合约,一个 Fabric 网络由多个 Peer 节点组成,每个 Fabric 节点上可携带多个账本,这些节点通过执行智能合约,可查询账本数据、验证交易,并与应用程序进行交互。用户输入的交易被引导到一个排序服务共识组件,由被称为排序服务节点的特殊节点验证交易,确保区块链的一致性,并将验证后的交易发送到网络中的 Peer 节点以及会员服务提供商(MSP)服务。除了私有通道功能外,Fabric 成员在一个联盟内可以相互交换私有数据,而不用通过 Fabric 通道传播,有助于保障数据安全。

2) Sawtooth 项目

Sawtooth 项目由 Intel 于 2016 年 4 月贡献社区,是第二个达到 1.0 版本成熟度的超级账本项目,该项目支持全新的基于硬件芯片的共识机制所用时间证明(PoET)。Sawtooth-core 采用 Python 编写,Sawtooth Raft 和 Sawtooth Sabre 采用 Rust 编写,拥有 JavaScript 和 Golang 组件。Sawtooth 支持有权限和无权限部署,通过与超级账本 Burrow 的合作支持 EVM。Sawtooth 项目通过将应用层与核心系统层分离,除提供基本功能外,还提供了诸如存储链上权限、管理全链设置,以及保存区块信息和性能分析等特殊应用。每个应用程序都可以定义自定义事务处理器,简化了应用的开发和部署,提供了智能合约抽象,允许开发人员用自己选择的编程语言编写合约。项目中的每个节点都可以通过验证和批准交易来充当排序器。该项目遵循先进的并行调度器,将交易分类为并行流,最终导致交易处理性能的提升。

3) Iroha 项目

Iroha 项目由 Soramitsu 等企业于 2016 年 8 月提交社区,该项目针对复杂数字资产和身份创建和管理设计,底层代码采用 C++ 编写。Iroha 项目基于角色进行访问控制,支持创建和管理复杂的数字资产并进行复杂的分析。当使用 Iroha 进行身份管理时,查询和执行命令只限于能够访问 Iroha 网络的参与者,权限系统确保了所有交易的安全和控制。Iroha 使用拜占庭容错共识算法,使用内置的智能合约(或称为“命令”)将区块链集成到业务流程中。

4) Indy 项目

Indy 项目由 Sovrin 基金会牵头研发,于 2017 年 3 月正式提交社区,该项目致力于打造一个基于区块链和分布式账本技术的数字身份管理平台。该项目中服务器部分 Indy

节点采用 Python 构建,Indy SDK 采用 Rust 编写,提供了工具和可重用的组件来管理区块链或其他分布式账本上的数字身份。

5) Besu 项目

Besu 项目是一个开源的以太坊(Ethereum)客户端,在 Apache 2.0 许可下开发,由 Consensus、Hacera、JMP 和 Red Hat 等公司发起支持,采用 Java 开发,于 2019 年 8 月正式贡献到社区。Besu 项目实现了企业以太坊联盟(EEA)规范,通过智能合约在以太坊虚拟机(EVM)环境中处理交易。在以太坊内的各种开源和闭源项目之间建立了共同的接口,确保了用户不会被供应商锁定,并为构建应用程序的团队创建标准接口。同时,Besu 实现了与 EEA 客户端规范一致的企业功能。Besu 项目支持多种共识算法,包括 PoW、PoA 和 IBFT,超级账本 Besu 还实现了多个 PoA 协议,包括 Clique 和 IBFT 2.0。

2. 开发工具

开源社区提供了大量易用的开发协作工具,主要有区块链应用测试、部署、管理等相关工具。下面简单介绍 Cello、Caliper 和 Avalon 三种开发协作工具。

1) Cello

Cello 项目由 IBM 于 2017 年 1 月提交社区。该项目的定位为区块链网络的操作系统,实现区块链网络自动化部署,以及对区块链网络的运行时间管理。使用 Cello 项目,可以让区块链应用人员专注于应用开发,而无须关心底层平台的管理和维护。Cello 项目的主要开发语言为 Python 和 JavaScript 等,底层支持包括裸机、虚拟机、容器云(包括 Swarm、Kubernetes)等多种基础架构。

2) Caliper

Caliper 项目由华为于 2018 年 3 月提交社区。Caliper 项目能够提供统一的工具套装评测区块链的吞吐、延迟、资源使用率等性能,主要基于 Node.js 语言实现,支持对 Fabric、Sawtooth、Burrow 等项目进行性能测试。

3) Avalon

Avalon 项目由 Intel、IEX、IBM 和 Consensus 等公司于 2019 年 9 月提交社区,该项目在设计上重点考虑了安全性和可扩展性,主要提供链下的安全计算支持,例如,帮助 Fabric 链码执行一些复杂的或安全性需求较高的计算。



视频讲解

5.2 超级账本浏览器

区块链浏览器与普通的浏览器不同,是建立在普通的中心化网络上,专门为用户提供浏览与查询区块链上信息的一种搜索工具,具有使用简便、功能强大、易维护等特点。

超级账本浏览器由 DTCC、IBM、Intel 等开发支持,是一个区块链的图形化浏览器,可通过浏览器查看网络中的信息,用于浏览底层区块链网络上的活动,包括网络拓扑、区块信息、链码信息等。目前支持 Fabric 和 Cello。

本章操作示例是在 Fabric 2.0 版本下部署,操作系统为 Ubuntu 18.04,部署搭建区块链浏览器可使用 Docker 容器或使用代码库,下面以 Docker 容器为例,源码地址为 <https://github.com/Hyperledger/blockchain-explorer>。

5.2.1 超级账本浏览器部署

1. 下载浏览器代码

打开上面提供的 GitHub 源码网址,如图 5.1 所示,单击 Code 按钮即可获取存储库代码位置。

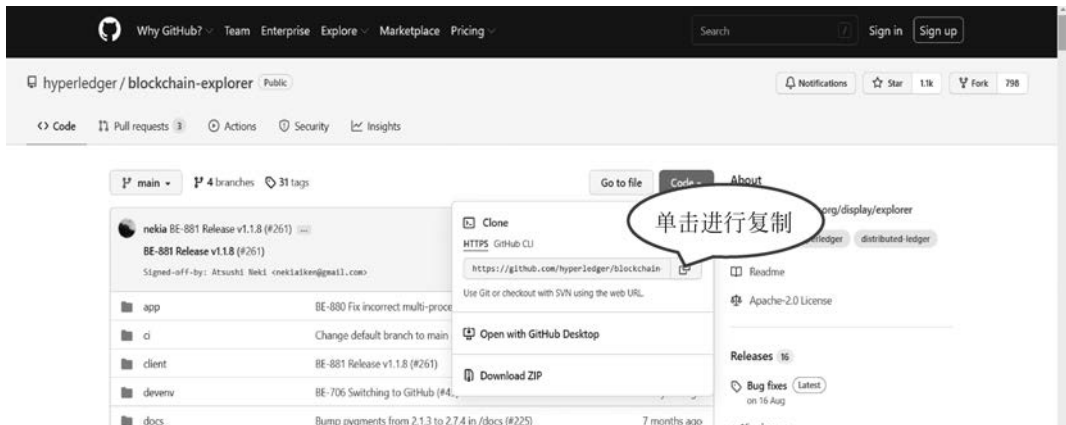


图 5.1 超级账本浏览器代码获取页面

使用如下命令克隆此存储库代码以获取最新版本。

```
1. git clone https://github.com/hyperledger/blockchain-explorer.git
```

2. 启动 first-network 网络

首先进入 first-network 文件夹,命令如下。

```
1. cd ~/go/src/github.com/hyperledger/Fabric-samples/first-network
```

启动 first-network 网络,命令如下。

```
1. ./byfn.sh up
```

若启动失败,尝试一下关闭 first-network 网络再启动,命令如下。

```
1. ./byfn.sh down
```

启动 first-network 网络详细内容请参阅第 7 章的内容。

3. 配置

首先,从存储库复制以下文件:

- docker-compose. yaml
- examples/net1/connection-profile/test-network. json
- examples/net1/config. json

复制文件命令如下。

```
1.  wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/
    examples/net1/config.json
2.  wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/
    examples/net1/connection-profile/test-network.json -P connection-profile
3.  wget https://raw.githubusercontent.com/hyperledger/blockchain-explorer/main/
    docker-compose.yaml
```

成功执行上述命令后,当前目录下有以下文件和目录结构。

- docker-compose. yaml
- config. json
- connection-profile/test-network. json
- organizations/ordererOrganizations/
- organizations/peerOrganizations/

编辑资源管理器容器文件(docker-compose. yaml),修改文件中 volumes 标签下的内容,使挂载在上面的网络名称及路径与环境保持一致。docker-compose. yaml 文件节选如代码清单 5-1 所示。

代码清单 5-1 docker-compose. yaml 文件

```
1.  networks:
2.      mynetwork.com:
3.          external:
4.              name: net_test
5.      ...
6.  services:
7.      explorer.mynetwork.com:
8.          ...
9.          volumes:
10.             - ./config.json:/opt/explorer/app/platform/Fabric/config.json
11.             - ./connection-profile:/opt/explorer/app/platform/Fabric/connection-profile
12.             - ../crypto-config/tmp/crypto
13.             - walletstore:/opt/explorer/wallet
```

为网络指定名称,更改密钥的值 name。

(1) 采用如下命令查询网络名称。

```
1.  docker network list
```

(2) 用查询到的网络名称替换 docker-compose. yaml 文件中 name 的值,如图 5.2 所示,第 12 行即为本文查询到网络名称后的替换结果,用“net_byfn”替换掉原名称“net_test”。

```

1
2 # SPDX-License-Identifier: Apache-2.0
3 version: '2.1'
4
5 volumes:
6   pgdata:
7   walletstore:
8
9 networks:
10  mynetwork.com:
11    external:
12    name: net_byfn
13

```

图 5.2 docker-compose. yaml 文件部分截图

(3) 修改路径使 volumes 路径与环境保持一致。修改图 5.3 中用方框框住的内容, 即 docker-compose. yaml 文件中 49~51 三行 volumes 路径。

```

48 volumes:
49   - ./examples/net1/config.json:/opt/explorer/app/platform/fabric/config.json
50   - ./examples/net1/connection:/opt/explorer/app/platform/fabric/connection-profile
51   - ./fabric-path/fabric-samples/test-network/organizations/tmp/crypto
52   - walletstore:/opt/explorer/wallet

```

图 5.3 修改路径前 docker-compose. yaml 文件截图

(4) 修改完成后, 如图 5.4 所示。用方框框住的内容是虚拟机路径, 后面是复制后启动 Docker 容器的路径。

```

48 volumes:
49   - ./config.json:/opt/explorer/app/platform/fabric/config.json
50   - ./connection-profile:/opt/explorer/app/platform/fabric/connection-profile
51   - ./crypto-config:/tmp/crypto
52   - walletstore:/opt/explorer/wallet

```

图 5.4 修改路径完成后 docker-compose. yaml 文件截图

(5) 在连接配置文件(test-network.json)中编辑管理证书和私有密钥的路径, 需要在资源管理器容器上指定绝对路径。如图 5.5 所示, 在 Fabric 2.0 和 2.2 版本中, 椭圆边框框住部分需要用“priv_sk”命名。

```

31 "adminPrivateKey": {
32   "path": "/tmp/crypto/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp/keystore/priv_sk"
33 },
34 "peers": [{"peer0.org1.example.com"},
35 "signedCert": {
36   "path": "/tmp/crypto/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp/signcerts/User1@org1.example.com-cert.pem"
37 }

```

图 5.5 test-network.json 文件截图

4. 启动容器服务

启动 Fabric 网络后, 执行以下命令启动 explore 和 explorer-db 服务。

```
1. docker - compose up -d
```

若启动失败, 运行结果如图 5.6 所示, 则进行第 5 步清理操作。

```

wp@wp-virtual-machine:~/go/src/github.com/hyperledger/fabric-samples/first-network/blockchain-explorer$ sudo docker-compose up -d
[sudo] wp 的密码:
Starting explorerdb.mynetwork.com ...
Starting explorer.mynetwork.com ... error
ERROR: for explorerdb.mynetwork.com Cannot start service explorerdb.mynetwork.com: network e908d3290acf167bdba363a8efed2e8095724b7915fe1ca7e8b8765167b35aa4 not found
ERROR: for explorerdb.mynetwork.com Cannot start service explorerdb.mynetwork.com: network e908d3290acf167bdba363a8efed2e8095724b7915fe1ca7e8b8765167b35aa4 not found
ERROR: Encountered errors while bringing up the project.

```

图 5.6 启动容器服务失败截图

5. 清理

(1) 停止业务而不移除持久数据,具体命令如下。

```
1. docker - compose down
```

(2) 命令成功执行后,在 docker-compose. yaml 文件中,为持久数据分配了两个 volumes(用于 Postgres 数据和用户钱包),再清除这些 volumes,运行如下命令。

```
2. docker - compose down - v
```

(3) 命令成功执行后,再回到第 4 步,启动容器服务。启动成功如图 5.7 所示。

```

wp@wp-virtual-machine:~/go/src/github.com/hyperledger/fabric-samples/first-network/blockchain-explorer$ docker-compose down
Removing explorerdb.mynetwork.com ... done
Network net_byfn is external, skipping
wp@wp-virtual-machine:~/go/src/github.com/hyperledger/fabric-samples/first-network/blockchain-explorer$ docker-compose down -v
Network net_byfn is external, skipping
Removing volume blockchainexplorer_pgdata
Removing volume blockchainexplorer_walletstore
wp@wp-virtual-machine:~/go/src/github.com/hyperledger/fabric-samples/first-network/blockchain-explorer$ sudo docker-compose up -d
Creating volume "blockchainexplorer_pgdata" with default driver
Creating volume "blockchainexplorer_walletstore" with default driver
Creating explorerdb.mynetwork.com ...
Creating explorerdb.mynetwork.com ... done
Creating explorer.mynetwork.com ...
Creating explorer.mynetwork.com ... done

```

图 5.7 启动容器服务成功截图

5.2.2 使用区块链浏览器

(1) 打开虚拟机浏览器,输入“localhost: 8080”网址进行查询,弹出如图 5.8 所示登录页面。



图 5.8 区块链浏览器登录页面

第一次登录需要输入账号密码,账号密码可在 test-network.json 文件中获取,如图 5.9 所示,本文账号(id)是“exploreradmin”,密码(password)是“exploreradminpw”。

```
1 {
2   "name": "test-network",
3   "version": "1.0.0",
4   "client": {
5     "tlsEnable": true,
6     "adminCredentia": {
7       "id": "exploreradmin",
8       "password": "exploreradminpw"
9     },
10    "enableAuthentication": true,
11    "organization": "Org1MSP",
12    "connection": {
13      "timeout": {
14        "peer": {
15          "endorser": "300"
16        },
17        "orderer": "300"
18      }
19    }
20  },
21 }
```

图 5.9 test-network.json 文件获取账号密码部分

(2) 登录即可进入区块链浏览器,如图 5.10 所示。

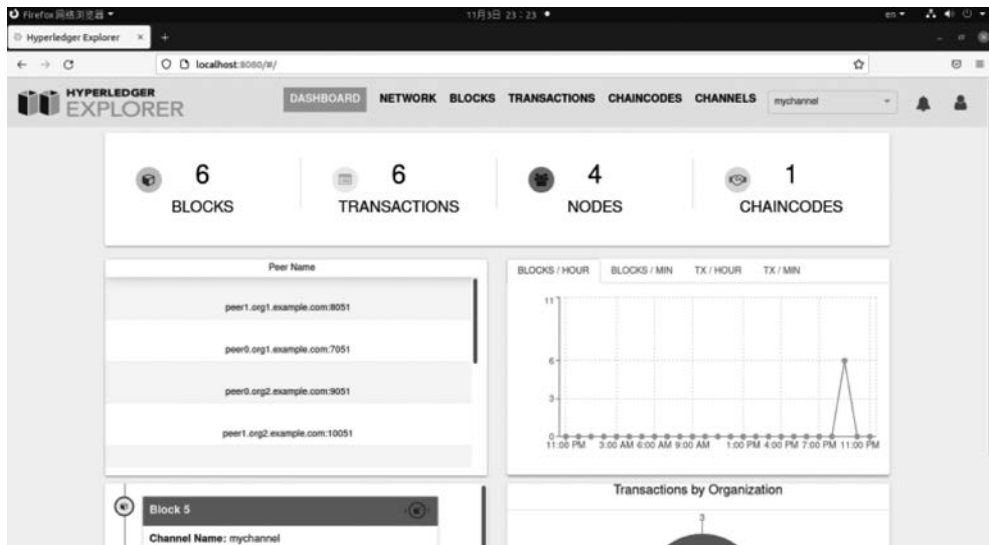


图 5.10 区块链浏览器界面

小结

本章从超级账本项目的起源、设计理念与特点、行业应用方面对超级账本项目进行了简单介绍,对超级账本子项目的特点与应用场景,以及超级账本提供的开发协作工具进行了概要性的介绍,重点讲解了超级账本区块链浏览器的使用方法。

希望读者完成本章学习后,对超级账本涉及的开源项目有一个初步的了解,能够总结超级账本开源子项目的差异与特点,并尝试画出知识架构图,并基于实际应用场景的需求,通过资源查找对相关内容进行更进一步的学习。

习题

- 5.1 请描述什么是超级账本 Fabric。
- 5.2 超级账本 Fabric 浏览器有什么用途？
- 5.3 Fabric 的特点有哪些？
- 5.4 超级账本下有哪些子项目？
- 5.5 你认为 Fabric 适用于什么样的应用场景？