IBM开源技术微讲堂

区块链和HyperLedger系列

第三讲

Hyperledger Fabric架构解读

http://ibm.biz/opentech-ma

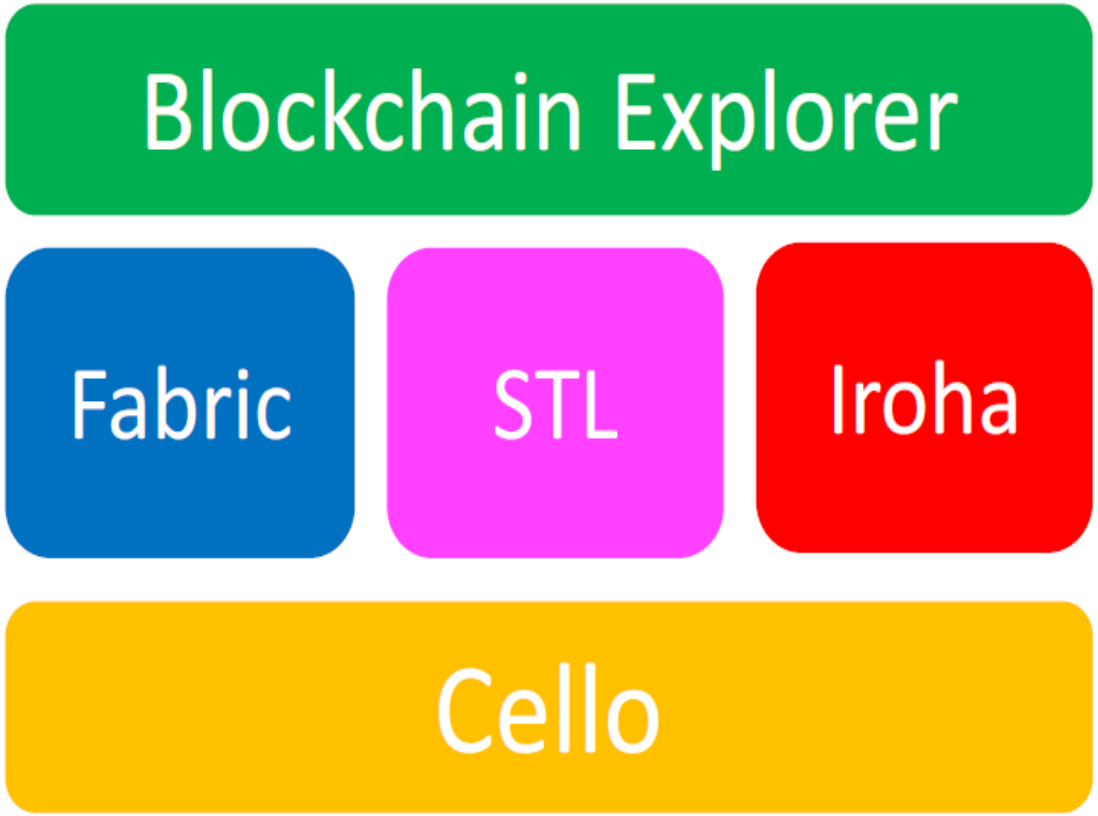
“区块链和HyperLedger”系列公开课

- 每周四晚8点档
  - 区块链商用之道
  - HyperLedger项目和社区概览
  - HyperLedger Fabric 架构解读
  - HyperLedger 中的共享账本
  - HyperLedger中的共识管理
  - HyperLedger中的隐私与安全
  - HyperLedger应用案例赏析

议程

- HyperLedger 子项目
- HyperLedger Fabric 0.6 架构
- HyperLedger Fabric 1.0 架构
- HyperLedger Fabric 1.0 环境搭建

HyperLedger 子项目



- **BlockChain Explorer**

  展示和查询区块链块、事务和相关数据的Web应用

- **Fabric**

  区块链技术的一个实现

- **STL - Sawtooth Lake**

  高度模块化的分布式账本平台

- **Iroha**

  轻量级的分布式账本， 侧重于移动

- **Cello**

  BaaS的工具集，帮助创建、管理、终止区块链

HyperLedger Fabric 0.6 逻辑架构

- 区块服务（BlockChain & Transaction）

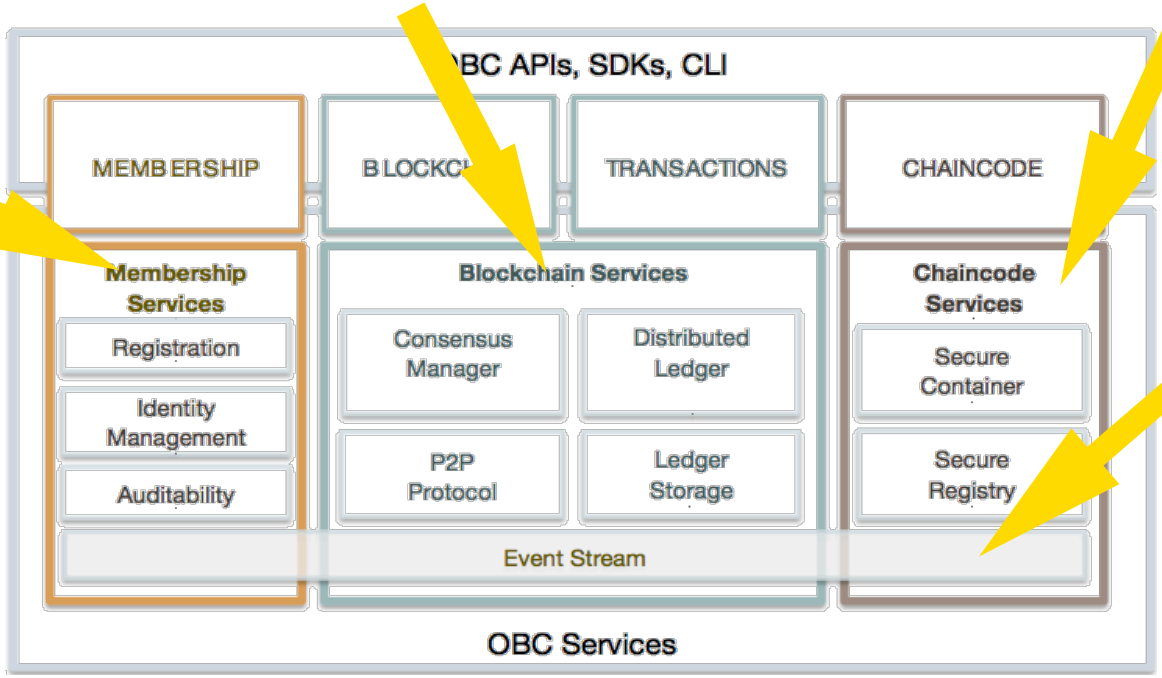
负责节点间的共识管理、账本的分布式计算、账本的存储以及节点间的P2P协议功能的实现，是区块链的核心组成部分，为区块链的主体功能提供了底层支撑。

- ChainCode

ChainCode的集成平台，为ChainCode提供安全的部署、运行的环境。

- 成员管理（Membership）

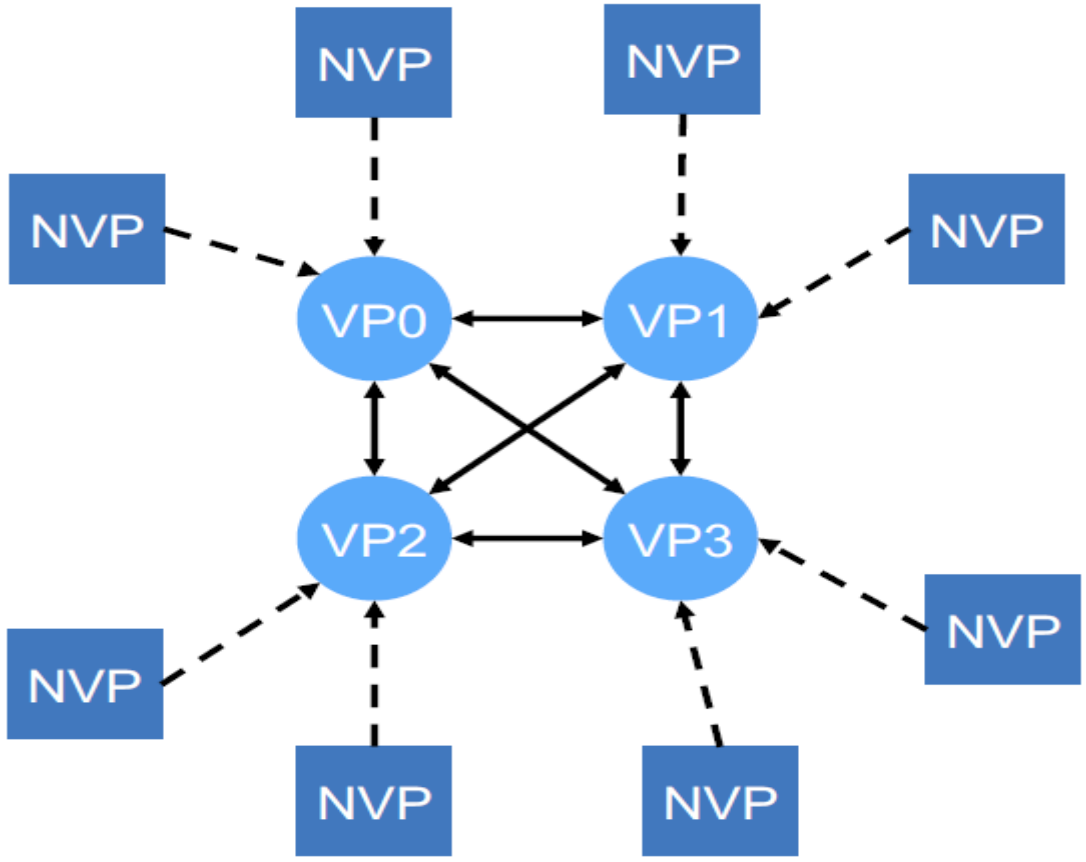
会员注册、身份保护、内容保密、交易审计功能，以保证平台访问的安全性。



- Event

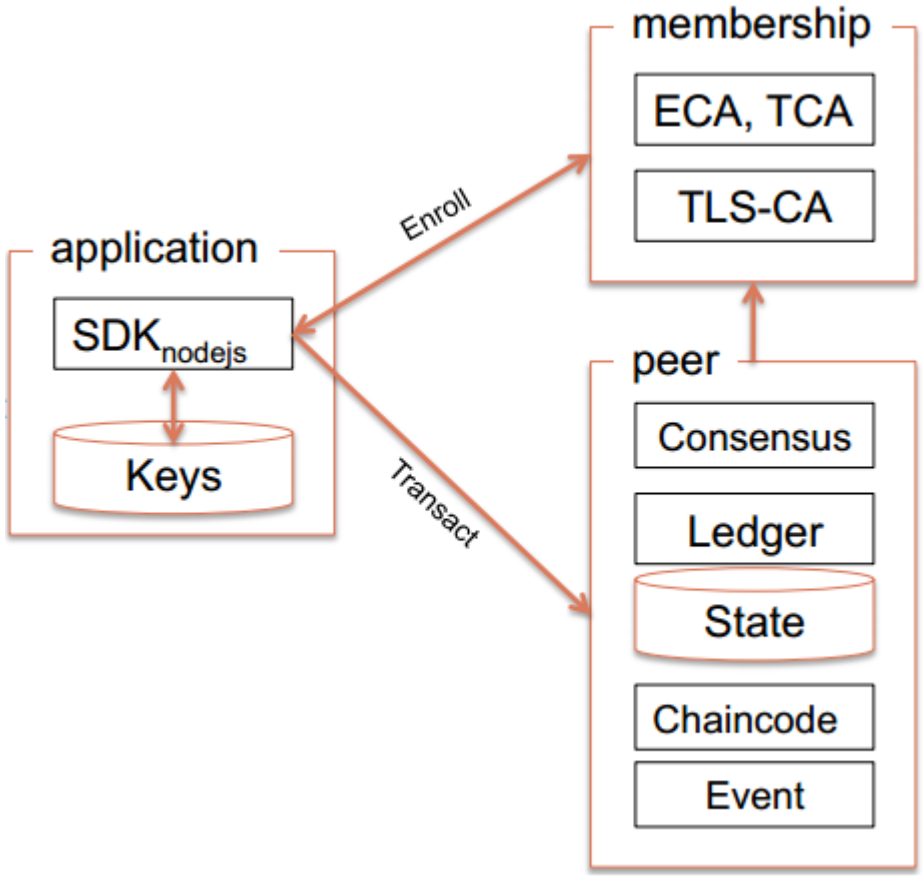
贯穿于其他各个组件中间，为各个组件间的异步通信提供了技术实现

HyperLedger Fabric 0.6 区块链网络



- 4个VP节点；
- 至多允许一个VP节点异常

HyperLedger Fabric 0.6 运行时架构



- 保证区块链的私有性，机密性，可审计性
- 可拔插的共识框架
  - PBFT, SIEVE (proto), NOOPS
- Chaincode 运行环境(Go, Java WIP)
  - Docker container (user-cc)
  - In peer process (system-cc)
- Client Node.js SDK
- REST APIs
- Basic CLI

HyperLedger Fabric 1.0 概述

目标

– 可伸缩性

– 性能

– 安全隔离

– 可拔插性

–可操作性

新功能

– 多通道

– 事务隔离（子账本）

– 可拔插的组件
  - 数据库
  - CA
  - 共识算法
  - …

– 更多类型的ChainCode

– ChainCode的版本

– Peer的高可用性

…

Hyerledger Fabric 1.0 逻辑架构

- 区块服务（BlockChain）

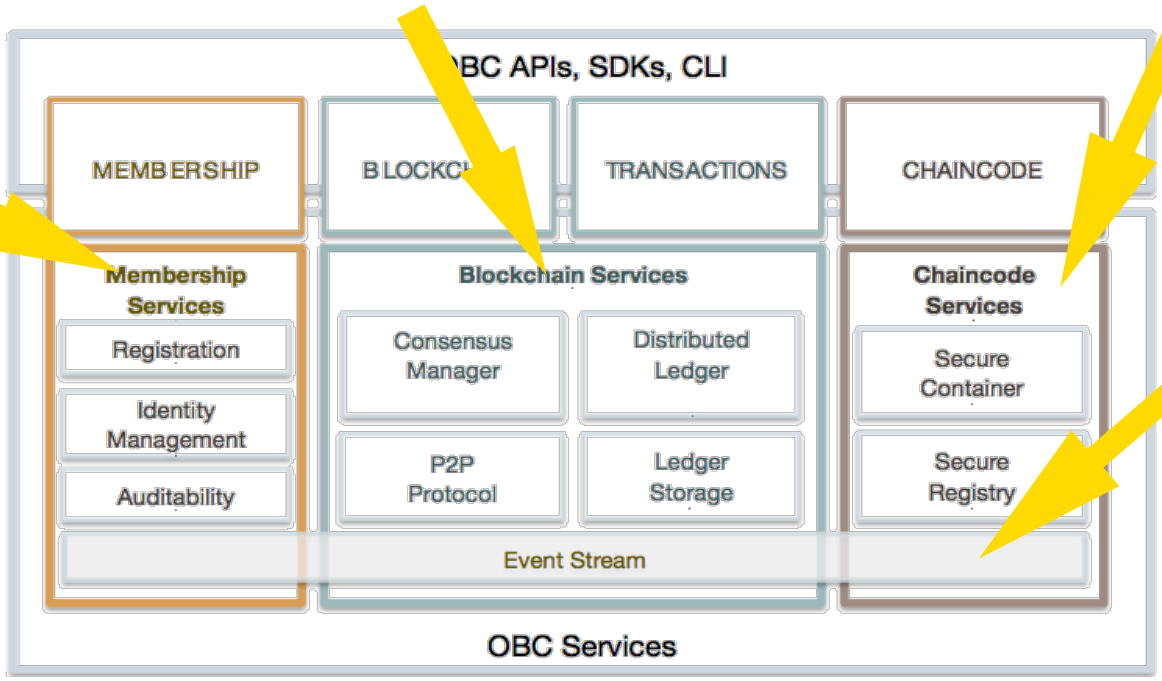
负责节点间的共识管理、账本的分布式计算、账本的存储以及节点间的P2P协议功能的实现，是区块链的核心组成部分，为区块链的主体功能提供了底层支撑。

- ChainCode

ChainCode的集成平台，为ChainCode提供部署、运行的环境。

- 成员管理（Membership）

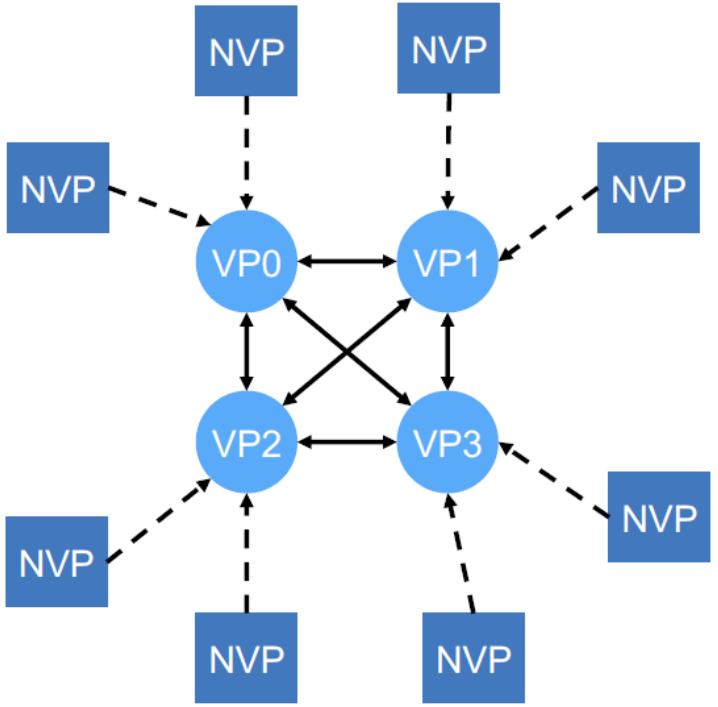
会员注册、身份保护、内容保密、交易审计功能，以保证平台访问的安全性。



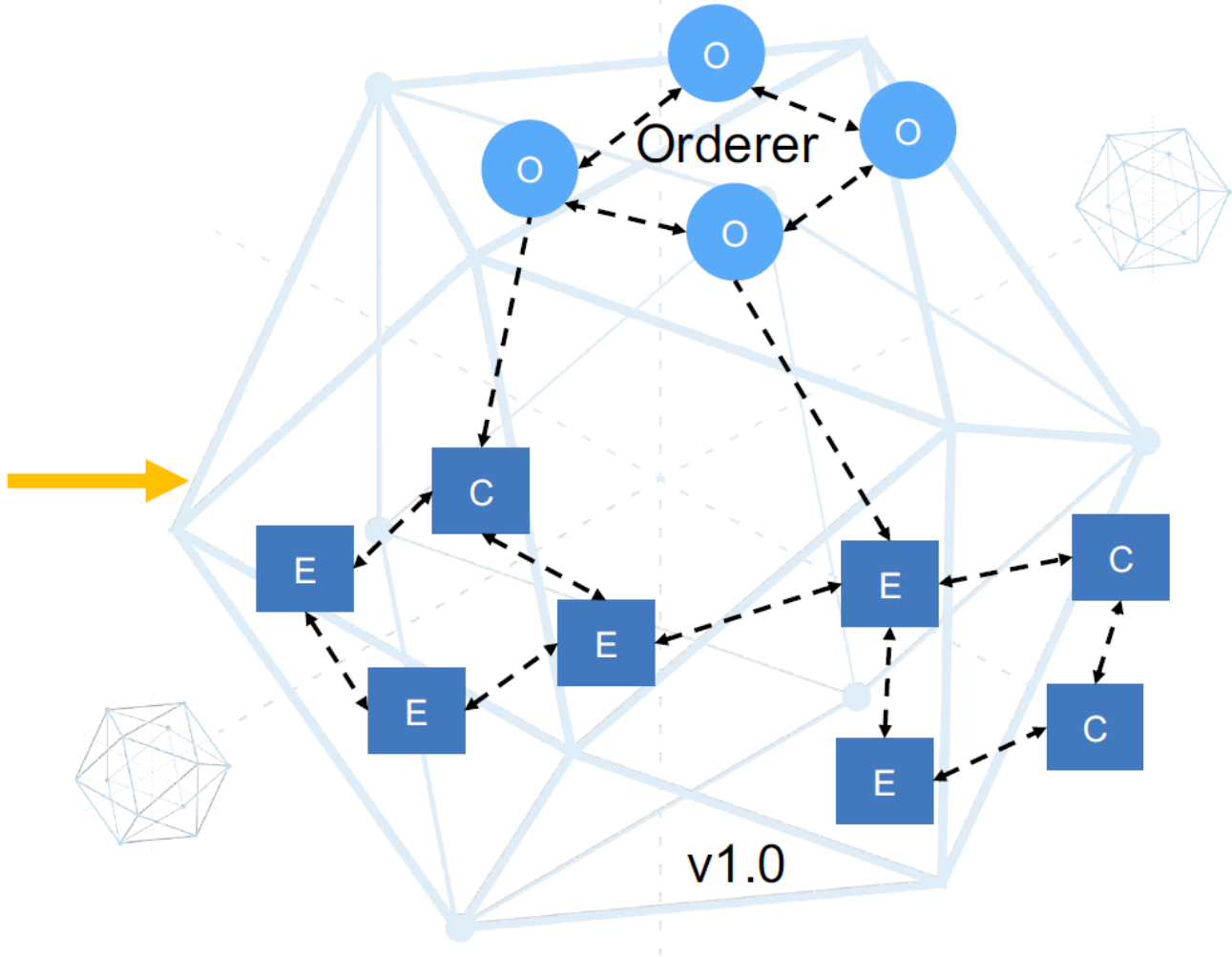
- Event

贯穿于其他各个组件中间，为各个组件间的异步通信提供了技术实现

HyperLedger Fabric 1.0 区块链网络

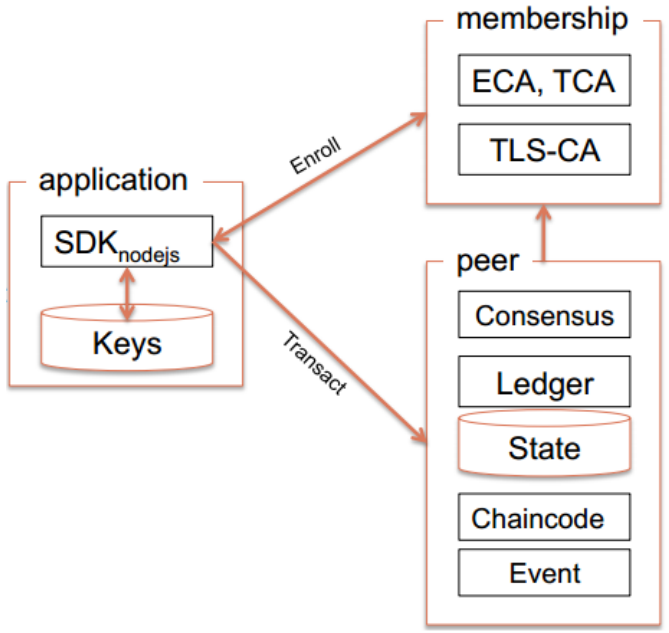


v0.6

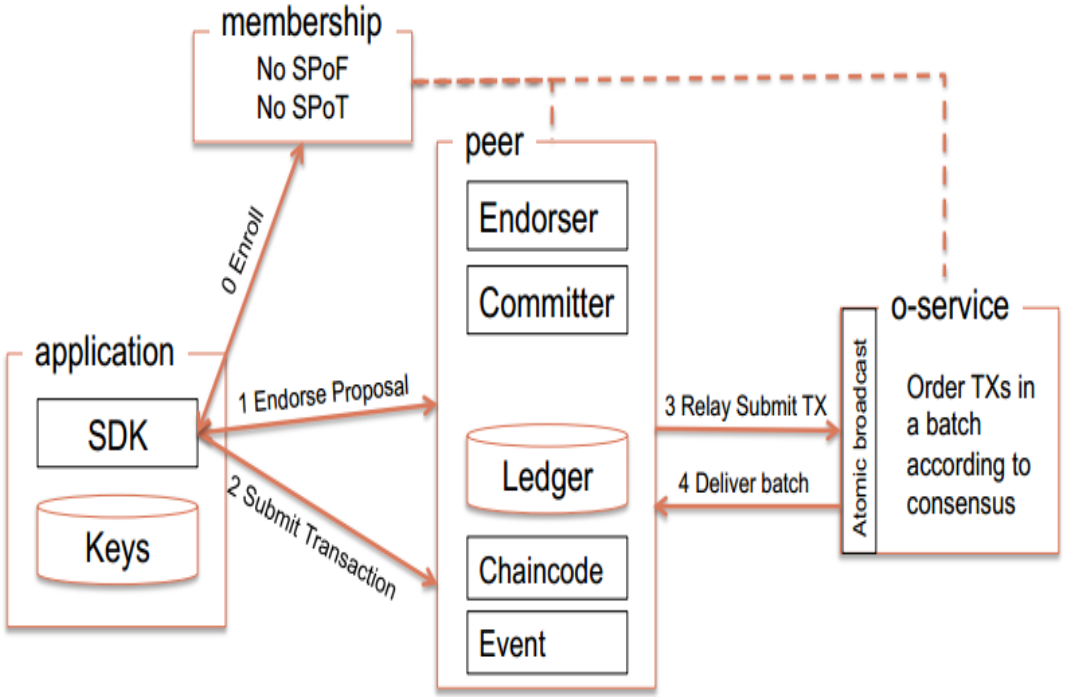


v1.0

HyperLedger Fabric 1.0 运行时架构

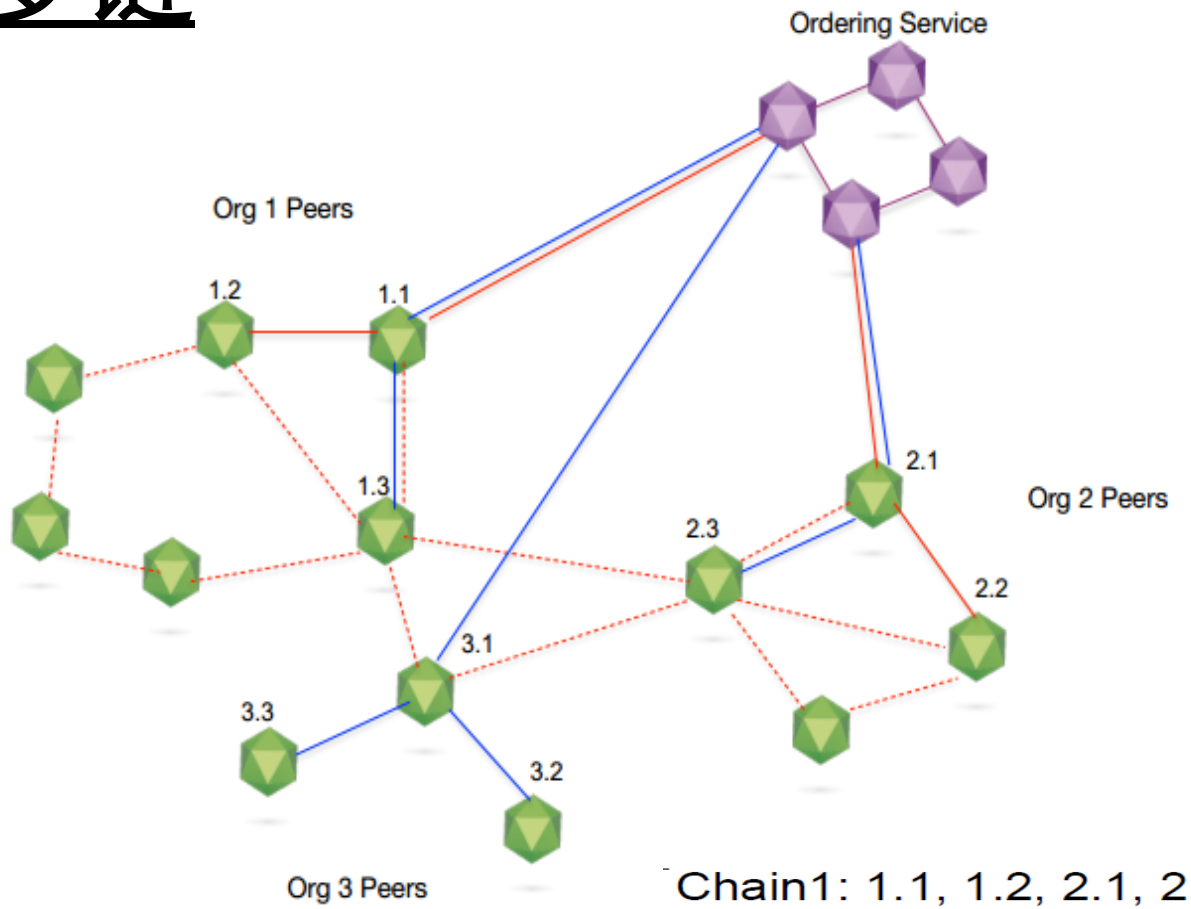


Fabric 0.6



Fabric 1.0

多链

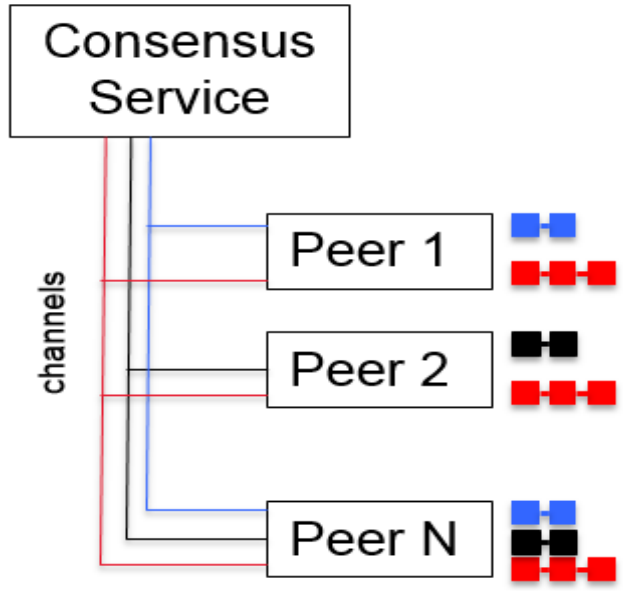


Chain1: 1.1, 1.2, 2.1, 2.2
Anchors: 1.3, 2,3
Leaders: 1.1, 2.1

Chain2: 1.1, 1.3, 2.2, 2.3, 3.2, 3.3

- 链=Peers + Ledger + Ordering Channel
- 链将参与者和数据（包含chaincode）进行隔离
- 一个Peer节点可以参与多个链

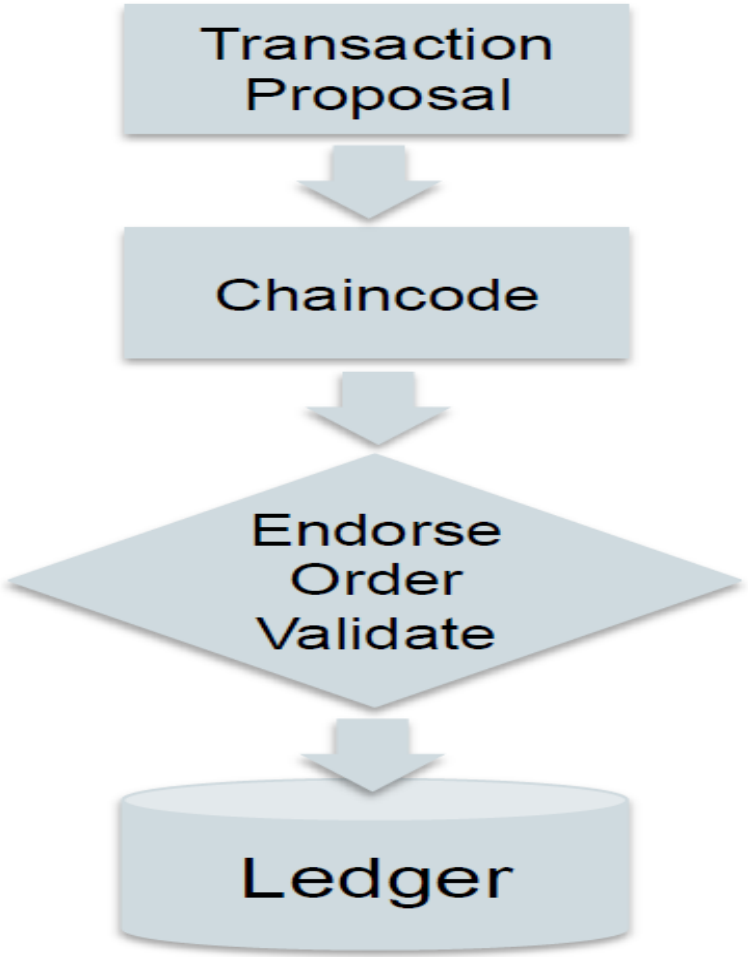
多通道 & 子账本



通道1（P1、 P2、P3 ）
通道2（P2、PN）
通道3（P1、P3）

- 通道： 通道提供一种通讯机制，将peer和orderer连接在一起，形成一个个具有保密性的通讯链路（虚拟）
- Fabric的区块链网络缺省包含一个账本（称为：系统账本）和一个通道；
- 子账本可以被创建，并绑定到一个通道

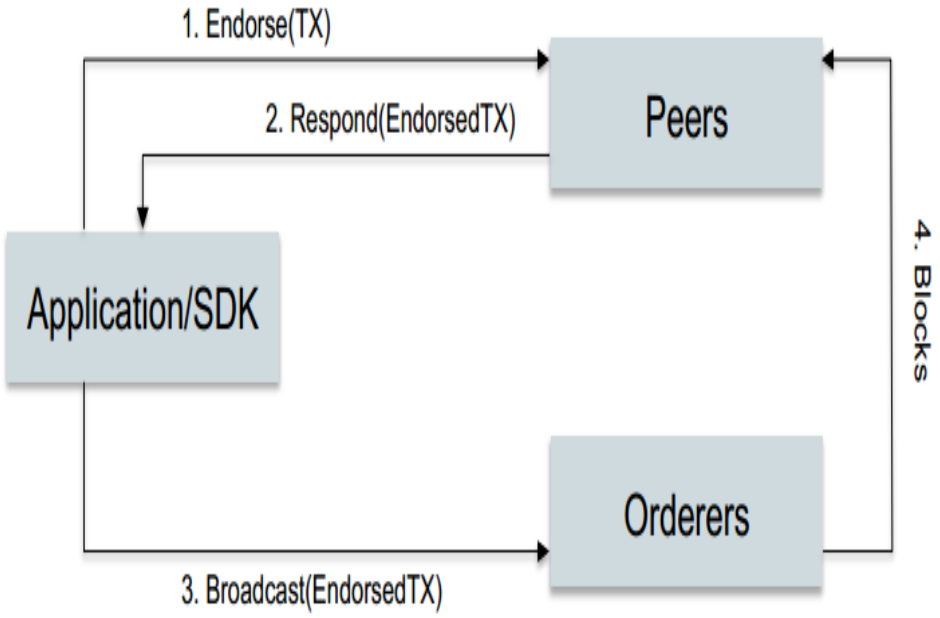
事务



- ChainCode的一次调用

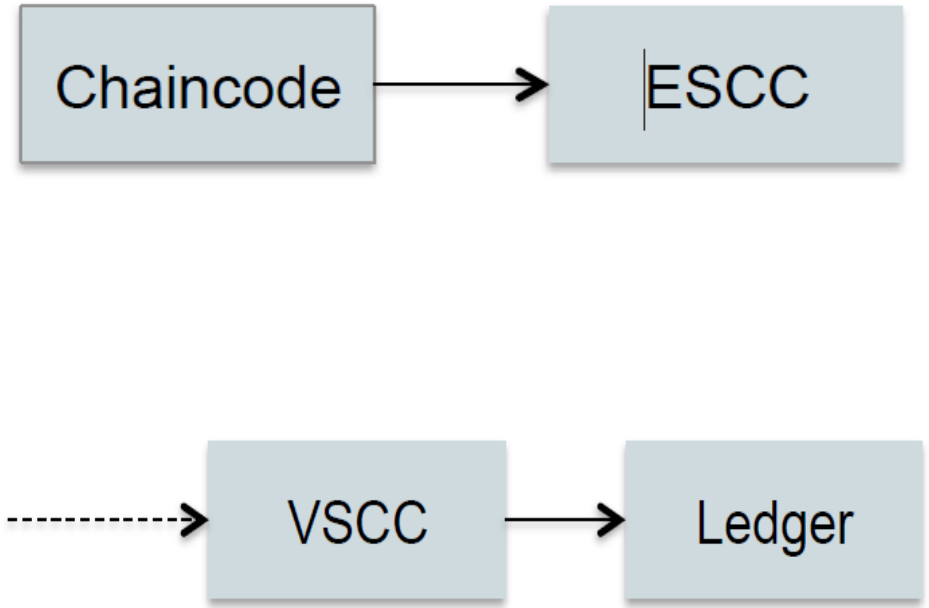
- 事务类型
  - Chaincode
  - Configuration
  - custom

事务处理流



- 应用向一个或多个Peer节点发送对事务的背书请求；
- 背书节点执行ChainCode，但并不将结果提交到本地账本，只是将结果返回给应用；
- 应用收集所有背书节点的结果后，将结果广播给Orderers；
- Orderers执行共识过程，并生成Block，通过消息通道批量的将Block发布给Peer节点；
- 各个Peer节点验证交易，并提交到本地账本中。

事务流（peer节点内）



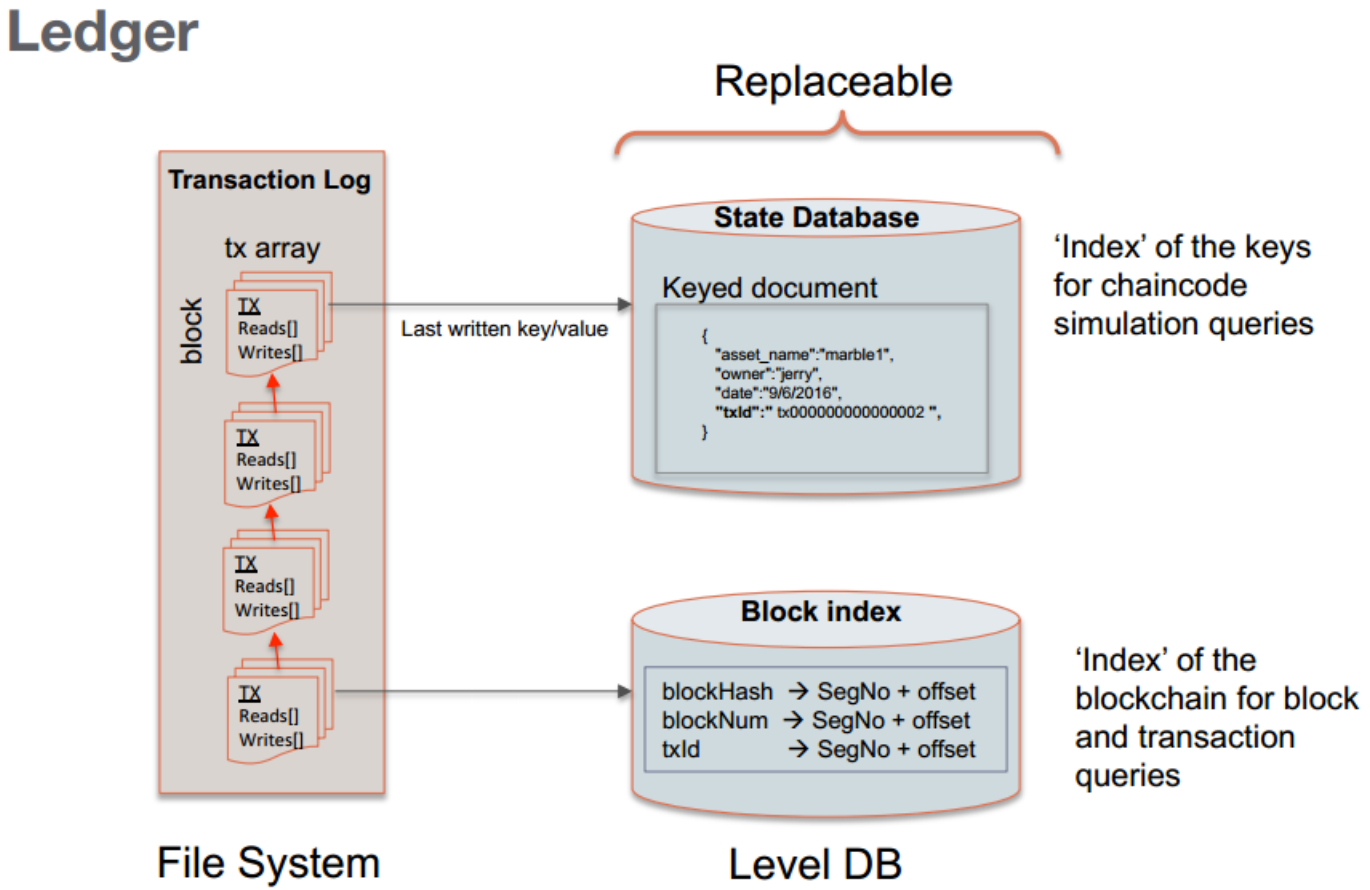
- 每个ChainCode在部署时，都需要和背书（ESCC）和验证（VSCC）的系统ChainCode关联；
- ESCC决定如何对proposal进行背书；
- VSCC决定事务的有效性（包括背书的正确性）；

:

IBM

Open by design™

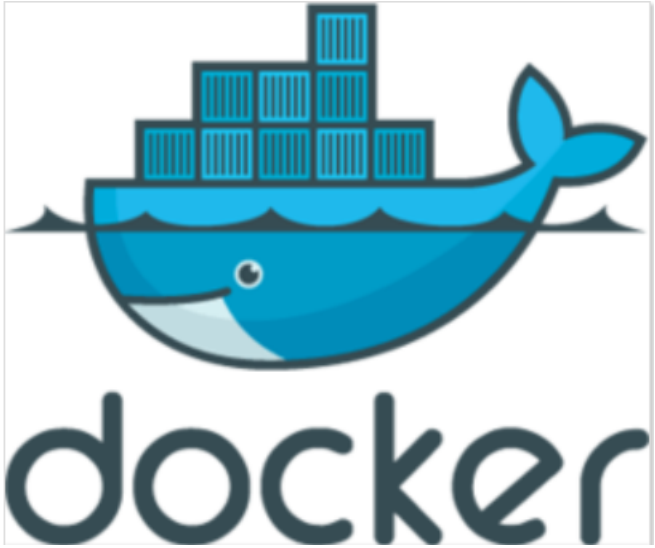
账本



- Block结构：文件系统方式存储
- State状态：KV数据库维护

HyperLedger Fabric Docker 安装

- Docker 1.12+
- Linux
  - 64 bit
  - kernel 3.10+
  - curl -sSL https://get.docker.com/ | sh
- Mac
  - Docker for Mac
- Docker-Compose 1.7.0+
  - pip install docker-compose>=1.7.0



- 修改Docker服务配置文件 （ /etc/default/docker 文件）

–DOCKER_OPTS="$DOCKER_OPTS -H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375”

- 重启Docker Daemon

sudo service docker restart

sudo systemctl restart docker

HyperLedger Fabric 1.0 的安装

- 下载Fabric 的Docker镜像

  官方镜像地址： https://hub.docker.com/r/hyperledger

获取Compose文件

- git clone https://github.com/yeasy/docker-compose-files

启动Fabric

- cd hyperledger/1.0
- docker-compose up

参考： http://hyperledger-fabric.readthedocs.io/en/latest/asset_setup.html

```
ca:
  image: hyperledger/fabric-ca
  container_name: fabric-ca
  hostname: ca
#  command: /go/src/github.com/hyperledger/fabric-ca/l
  ports:
    - "8888:8888"
  command: fabric-ca server start -ca testdata/ec.pem
```

```
orderer:
  image: hyperledger/fabric-orderer
  container_name: fabric-orderer
  hostname: orderer
  environment:
      - ORDERER_GENERAL_LEDGERTYPE=ram
      - ORDERER_GENERAL_BATCHTIMEOUT=10s
      - ORDERER_GENERAL_MAXMESSAGECOUNT=10
      - ORDERER_GENERAL_MAXWINDOWSIZE=1000
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_LISTENPORT=7050
      - ORDERER_RAMLEDGER_HISTORY_SIZE=100
      - CONFIGTX_ORDERER_ORDERERTYPE=solo
  ports:
    - "7050:7050"
  command: orderer
```

```
peer0:
  extends:
    file: peer.yml
    service: peer
  container_name: fabric-peer0
  hostname: peer0
  environment:
    - CORE_PEER_ID=peer0
    - CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer:7050
  links:
    - ca
    - orderer
  ports:
    - 7051:7051
  depends_on:
    - ca
    - orderer
```

IBM

Open by design™

IBM开源技术微讲堂

区块链和HyperLedger系列

第三讲完

http://ibm.biz/opentech-ma