# HyperLedger：超级账本

Hyperledger (或 Hyperledger项目)是一个旨在推动区块链跨行业应用的开源项目， 由 Linux基金会在2015年12月主导发起该项目， 成员包括金融，银行，物联网，供应链，制造和科技行业的领头羊。

**Hyperledger的区块链平台有：**

Hyperledger Burrow

Hyperledger Fabric：区块链技术的一个实现

Hyperledger Iroha：轻量级分布式账本，侧重于移动

Hyperledger Sawtooth：高度模块化的分布式账本平台

Hyperledger Indy

请访问官网：<https://www.hyperledger.org/>

请访问社区：<https://www.hyperledger.org/members>

**Hyperledger Burrow**

Burrow是一个包含了“built-to-specification”的以太坊虚拟机.区块链客户端。其主要由Monax贡献，并由Monax 和英特尔赞助。Burrow是最早Monax开发的项目，后来进入HyplerLedger孵化。Burrow的Incubation proposal中介绍，burrown之前的名字是 eris-db。

**Hyperledger Fabric**

Hyperledger Fabric是一个许可的区块链构架(permissioned blockchain infrastructure)。其由IBM和Digital Asset最初贡献给Hyperledger项目。它提供一个模块化的构架，把架构中的节点、智能合约的执行(Fabric项目中称为"chaincode") 以及可配置的共识和成员服务. 一个Fabric网络包含同伴节点（"Peer nodes"）执行chaincode合约，访问账本数据，背书交易并称为应用程序的接口。命令者节点（"Orderer nodes"）负责确保此区块链的一致性并传达被背书的交易给网络中的同伴们；以及MSP服务，主要作为证书权威（Certificate Authority）管理X.509证书用于验证成员身份以及角色。

**Hyperledger Iroha**

Iroha是一个基于Hyperledger Fabric主要面向移动应用的协议，由日本的金融科技公司Soramitsu贡献。

**Hyperledger Sawtooth**

由Intel贡献的Sawtooth利用一种新型公式机制称为时间流逝证明（"Proof of Elapsed Time,"）一种基于可信的执行环境的彩票设计模式的共识协议由英特尔的Software Guard Extensions (SGX)提供。相比Fabric，Sawtooth的一个亮点是能够并行的处理事务：支持事件订阅：

# 区块链工具：

* **Cello：**是一个部署管理平台，也就是现在常说的baas，用来管理组成链的节点，以及部署链、管理链。用户可以直接通过Cello部署合约。
* **Composer：**是用于合约、区块链应用开发的工具，用来简化、加快区块链应用的开发过程。
* **Explorer：**是一个区块链浏览器。
* Caliper是一个Benchmark工具，华为贡献的，Measuring Blockchain Performance with Hyperledger Caliper。
* Quilt是Interledger Protocol (ILP)协议的Java实现，是日本的NTT Data贡献的。 日本人对区块链很热情嘛，10个项目中，有2个是日本公司贡献的。Interledger Protocol (ILP)定义了分布式账本与分布式账本之间、传统账本与分布式账本之间的交互过程。

# Hyperledger Fabric概述

Hyperledger Fabric是由IBM公司主导开发的一个面向企业级客户的开源项目。与比特币和以太坊这类公有链不同，Hyperledger Fabric网络中的节点必须经过授权认证后才能加入，从而避免了POW资源开销，大幅提高了交易处理效率，满足企业级应用对处理性能的诉求。在permissioned networks（授权网络）上，进行private confidential transactions（私人保密交易）。Fabric的定位是商业应用、是企业与企业之间的联盟链。联盟链是不能随随便便接入的，必须有准入限制（permissioned）。相比较于比特币的网络是全公开的网络，任何人都可以接入。同时，为了满足灵活多变的应用场景，Hyperledger Fabric采用了高度模块化的系统设计理念，将模块进行分离部署，使开发者可以根据具体的业务场景替换模块，实现了模块的插件式管理（plug-in/plug-out）。所以，Hyperledger Fabric是一个私有链／联盟链的开发框架，而且系统的运行不需要token支持。

* 权限认证模块（MSP）、
* 共识服务模块（Ordering Service）、
* 背书模块（Endorsing peers）：背书是指持票人为将票据权利转让给他人或者将一定票据权利授予他人行使。
* 区块提交模块（committing peers）

**关键组件：**

* Channel(通道)：又称子链，是一种数据隔离机制，保证交易信息只有交易参与方可见，每个channel是一个独立的私有区块链，这使得多个用户可以共用同一个区块链系统而不用担心信息泄露问题。同一peer可加入不同channel，chaincode的操作基于channel进行，同一channel上的peer节点同步其上chaincode执行的结果
* Chaincode：链码、链上代码，也叫智能合约，将资产定义和资产处理逻辑封装成接口，当其被用户调用的时候，改变账本的状态。生成Transaction(交易)的唯一来源，接口的实现代码，部署在fabric区块链的peer(网络节点)上，与fabric区块链交互的唯一渠道，智能合约在fabric上的实现方式
* Ledger：区块链账本，保存交易信息和智能合约代码。是一个channel(通道)的chain,由channel中每个peer维护的数据库。
* Network：交易处理节点之间的P2P网络，用于维持区块链账本的一致性。
* World state：显示当前资产数据的状态，底层通过LevelDB和CouchDB数据库将区块链中的资产信息组织起来，提供高效的数据访问接口。
* Membership service provider（MSP）：管理认证信息，为client和peers提供授权服务。为client和peer提供证书的系统抽象组件，Client用证书来认证他们的交易；peer用证书认证其交易背书。该接口与系统的交易处理组件密切相关，旨在使已定义的成员身份服务组件以这种方式顺利插入而不会修改系统的交易处理组件的核心。
* Ordering Service ：排序服务或共识服务，独立于peer流程之外，以先到先得的方式为网络上所有的channel做教育排序，支持可插拔实现，目前默认Solo（单节点共识）、kafka（分布式队列）和SBFT（简单拜占庭容错）。是整个网络的公用binding，包含与每个Member（成员）相关的的加密材料。利用kafka、SBTF等共识算法对所有交易信息进行排序并打包成区块，发给committing peers节点(提交节点)，写入区块链中。

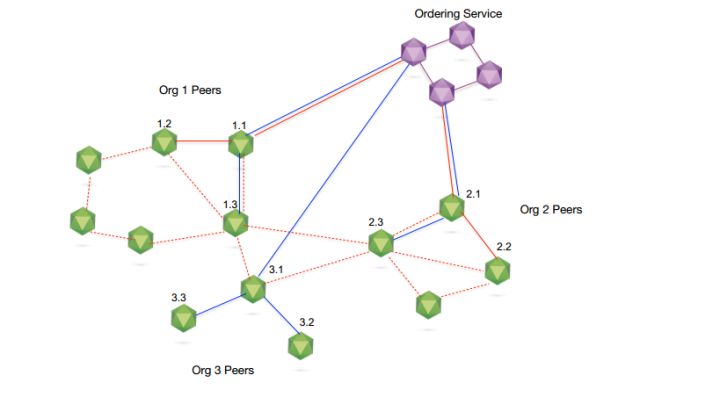
**Hyperledger Fabric Network中的由三种类型的角色：**

* Client：应用客户端，用于将终端用户的交易请求发送到区块链网络；
* Orderer:共识服务认证，对chaincode执行结果consensus。支持solo/kafka/sBFT不同的ordering策略
* Peers：负责维护区块链账本，分为endoring peers和committing peers，其中，endorser为交易做背书（验证交易并对交易签名），committer接收打包好的区块，然后写入区块链中。Peers节点是一个逻辑的概念，endorser和committer可以同时部署在一台物理机上。
* Ordering Service：接收交易信息，并将其排序后打包成区块，放入区块链，最后将结果返回给committer peers。

endorser:执行chaincode分离计算任务，减轻consensus节点负担，增加吞吐量，支持endorsement policy ，更加灵活。

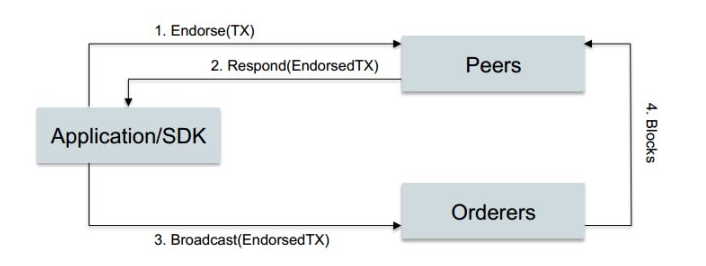
Committer:将chaincode执行结果写入ledger(区块链账本)

endorser:背书人 order:秩序 orderer:订货人 consensus:共识



**Hyperledger交易流程**

1. 客户端通过SDK接口，向endorsing peer节点(背书节点)发送交易信息：
2. 每个endorsing peer节点模拟处理交易，此时并不会将交易信息写入账本。然后，endorser peer会验证交易信息的合法性，并对交易信息签名后，返回给client。此时的交易信息只是在client和单个endorser peer之间达成共识，并没有完成全网共识，各个client的交易顺序没有确定，可能存在双花问题，所以还不能算是一个“有效的交易”。同时，client需要收到“大多数”endorser peer的验证回复后，才算验证成功，具体的背书策略由智能合约代码控制，可以由开发者自由配置。
3. client将签名后的交易信息发送给order service集群进行交易排序和打包。Order service集群通过共识算法，对所有交易信息进行排序，然后打包成区块。Order service的共识算法是以组件化形态插入Hyperledger系统的，也就是说开发者可以自由选择合适的共识算法，共识算法有：Solo（单节点共识）、kafka（分布式队列）和SBFT（简单拜占庭容错）。
4. ordering service将排序打包后的区块广播发送给committing peers，由其做最后的交易验证，并写入区块链。ordering service只是决定交易处理的顺序，并不对交易的合法性进行校验，也不负责维护账本信息。只有committing peers才有账本写入权限。
5. 总体交易流程图（待自己完善）



State状态：

KV数据库维护

Block结构：

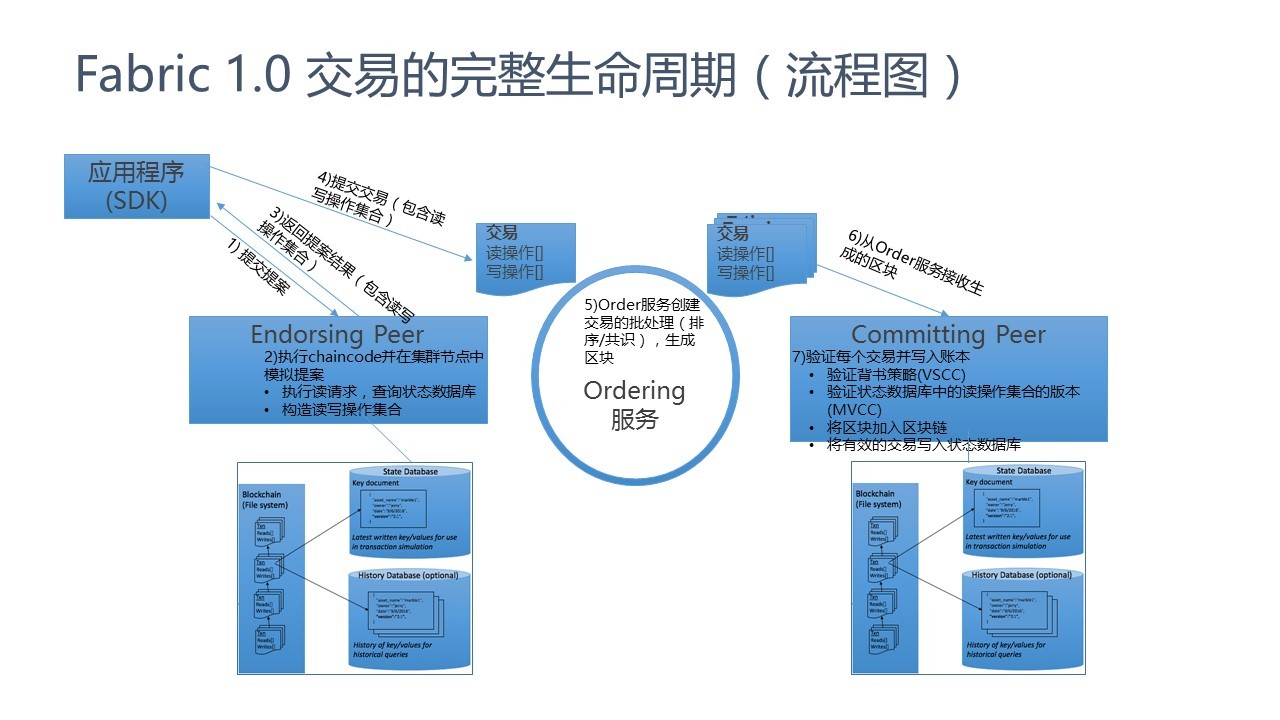
文件系统方式存储

File system

Level DB

**endorsing peer**

**committing peer**





**交易流程总结**

区块链的账本由peer节点维护，并不是由ordering service集群维护，所以，只有peer节点上可以找到完整的区块链信息，而order service集群只负责对交易进行排序，只保留处理过程中的一部分区块链信息。Hyperledger Fabric系统中的节点是一个逻辑的概念，并不一定是一个台物理设备，但是对于生产环境的设计者来说，peer节点不能和order节点部署在一台机器上，而enduring peers和committing peers可以部署在同一台机器上，这种设计主要是为了系统架构的解耦，提高扩展性，以及通过主机隔离提高安全性。 Endorsing peer校验客户端的签名，然后执行智能合约代码模拟交易。交易处理完成后，对交易信息签名，返回给客户端。客户端收到签名后的交易信息后，发给order节点排序。Order节点将交易信息排序打包成区块后，广播发给committing peers，写入区块链中。

**Hyperledger Fabric Network的共识算法：**

在所有peers中，交易信息必须按照一致的顺序写入账本（区块链的基本原则）。例如，比特币通过POW机制，由最先完成数学难题的节点决定本次区块中的信息顺序，并广播给全网所有节点，以此来达成账本的共识。而Hyperledger Fabric采用了更加灵活、高效的共识算法，以适应企业场景下，对高TPS的要求。目前，Hyperledger Fabric有三种交易排序算法可以选择。

* SOLO：只有一个order服务节点负责接收交易信息并排序，这是最简单的一种排序算法，一般用在实验室测试环境中。Sole属于中心化的处理方式。
* Kafka：是Apache的一个开源项目，主要提供分布式的消息处理／分发服务，每个kafka集群由多个服务节点组成。Hyperledger Fabric利用kafka对交易信息进行排序处理，提供高吞吐、低延时的处理能力，并且在集群内部支持节点故障容错。
* SBFT：简单拜占庭算法，相比于kafka，提供更加可靠的排序算法，包括容忍节点故障以及一定数量的恶意节点。目前，Hyperledger Fabric社区正在开发该算法。

**Channel的概念**

Channels能够让上层不同的用户业务共享同一个区块链系统资源，主要包括网络、计算、存储资源。从本质上来说，channels是通过不同的区块链账本来为上层业务服务，而且，这些区块链统一部署在peers节点上，统一通过ordering service进行交易排序和打包区块。Channels之间通过权限隔离控制，不同channel内的成员，无法访问对方的交易信息，只能访问所属channel的交易信息。

channel可以理解为系统资源的逻辑单元，每个channel都包含peers资源、order资源、网络资源等等，而且这些资源有可能是和其它channel所共享。

**State Database**

状态数据库保存了账本所有资产的最新状态（例如，账户A拥有某种资产的总量），同时，为智能合约提供了丰富的资产查询语义。所有的资产信息最终以文件形式记录在区块链账本中，而数据库是区块链账本的视图表现形式，能够让智能合约更加高效的和账本信息进行交互。数据库自动从底层区块链账本中更新或者恢复数据，默认的状态数据库是LevelDB，也可以替换为CouchDB。

LevelDB：Hyperledger Fabric的默认数据库，简单的存储键值对信息；

CouchDB：提供更加丰富的查询语义，可以保存JSON对象，以及范围key的查询。

**Smart Contract**

智能合约就是一段部署在区块链账本中的计算机程序，用于执行交易以及修改资产的状态。在Hyperledger Fabric中，智能合约被称作chaincode，使用Go语言编写。

**Membership Service Provider（MSP）**

Hyperledger Fabric是一种permissioned blockchain，所有的节点都是必须经过授权后才能访问区块链网络（比特币属于permissionless blockchain）。MSP是Hyperledger Fabric中的身份认证模块，用于对用户身份的校验、授权以及网络访问权限控制。默认的MSP接口是Fabric-CA API，同时，开发者可以根据自身的业务需要，实现自己的身份认证接口，对接MSP。Hyperledger Fabric网络可以被多个MSP控制，用以满足各个组织的需要。

**Hyperledger Fabric的商业价值**

随着比特币、以太坊等一系列虚拟货币的疯涨，区块链一度被认为是可以颠覆互联网的下一代革命性技术。但是，我们需要清楚的意识到，虚拟货币的价格并不能代表区块链技术能够为人类创造的实际价值，而且，大多数区块链应用本身并不需要token激励机制。所以，区块链未来的发展取决于能否广泛应用在商业领域，为人类的生活生产降低成本、提高效率。

Hyperledger Fabric的意义在于，迈出了区块链向商业领域进军的第一步，未来会有更多的区块链项目在各个行业中创造价值。

**Fabric 0.6与1.0**

Fabric 0.6的特点：

* 结构简单： 应用-成员管理-Peer的三角形关系，主要业务功能全部集中于Peer节点；
* 架构问题：由于peer节点承担了太多的功能，所以带来扩展性、可维护性、安全性、业务隔离等方面的诸多问题，所以0.6版本在推出后，并没有大规模被行业使用，只是在一些零星的案例中进行业务验证

Fabric 1.0的特点：

* 分拆Peer的功能，将Blockchain的数据维护和共识服务进行分离，共识服务从Peer节点中完全分离出来，独立为Orderer节点提供共识服务；
* 基于新的架构，实现多通道（channel）的结构，实现了更为灵活的业务适应性（业务隔离、安全性等方面）
* 支持更强的配置功能和策略管理功能，进一步增强系统的灵活性和适应性；

**fabric结构**

Fabric组件只有两个：Peer和Orderer，它们就是两个二进制程序。每个参与方可以选择部署一个/多个Peer，或者一个/多个Orderer，或者两个都部署。所有参与方的Peer/Orderer，彼此通信、连接，就组成了Fabric网络。

* Peer：每个Peer中都存放全量的数据（账本），也就是完整的链（必须是Peer参与的链，见后面“控制信息的传播范围”）。
* Orderer：通过Orderer之间的协商，确保每个Peer都能收到正确的数据。
* system chain：Fabric中有一组特殊的链，叫做“system chain”。这组链相当于整个Fabric网络的配置文件，里面记录了所有的Channel信息、参与者的信息。（每个Channel对应一条system chain）
* 创世块（Gengsis Block）：是system chain的第一个区块。这个区块通常是用configtxgen命令生成的。
* Anchor Peer(锚点):每个参与者可以将自己的一个Peer地址写入到system chain中，这样的Peer被称为“锚点”（Anchor Peer）。Orderer们从system chain中获得Anchor Peer的地址，并将形成的共识通知给它们。Anchor Peer通过Gossip协议，将结论八卦给其它的Peer。通道中能被所有对等节点探测，并能与之通信的一种对等节点。通道中都有一个或多个锚节点，允许属于不同成员身份的节点来发现通道中存在的其他节点。
* Leader Peer：Orderer是将共识发送给每个组织的Leader Peer，Leader Peer是一个组织的多个Peer之间选举出来的，或者组织的主动指定的。Anchor Peer（锚点）主要作用是用于Peer之间的发现。
* System Chain与Orderer的存在，使“控制信息的传播范围”成为可能。既然Orderer是输出数据的源头，那么就可以通过Orderer控制信息的传播范围了。
* 联盟链：企业与企业之间的链，不全公开，接入需要验证接入。
* 私链：也就是Channel，Orderer的data目录，就会发现每个Channel都会有一个独立的链。

Peer的data目录中，只会有Peer所加入的Channel的链：

Orderer中的链与Peer中的链是不同的，Oderer的链中存放的是Channel的配置，是system chain，Peer的链中存放的是Channel的数据。比对一下它们的内容就可以知道。

Peer与Orderer交织，数据更新请求在Orderer之间来回穿梭，最终被送往了Anchor Peer，大嘴巴的Anchor Peer转身通知了身边的所有Peer。

**合约**

* 参与者们在现实世界里达成共识。这个共识，更像谈判后签署的合同，在区块链中，被称为合约。
* 落实到Fabric，合约就是分布在每个Peer上的容器。
* 首先使用Go或者js编写处理逻辑，代码中约定了数据的修改方式。然后这些代码会被打包、签署，并提交到Fabric中，最终成为Peer上的一个Docker容器。
* 合约是可以升级、改动的,只有在当初签署了合约的人员都同意的情况下，才能更改合约。
* 背书策略的设计，通过背书的策略，可以约定只有当多个Peer上的合约都得出同样的结果时，对合约的调用才能被接受，只对自己的Peer动手脚是不行的。
* 合约需要在每个Peer上进行安装的，但激活只需要一次。安装合约其实就是将打包的合约上传到了Peer上（当然这中间还有一些其它检查）：

**MSP**

* 它是一个包含了根证书、证书（经过CA签署的公钥）和私钥的目录,在组织、组件和用户，三个地方会用到。
* 组织的msp目录中包含的全是可以公开的证书，没有私钥。因组织的msp目录,是要被写入到system chain中的,被包含在组织msp目录中的证书，主要被用来验证用户证书的有效性。
* 用户的证书的有效性：必须是用根证书对应的私钥签署的，否则就会被认定为无效，并拒绝服务。
* 组件msp目录的使用者是Fabric网络中的peer和orderer程序
* 用户msp目录的使用者是Fabric网络之外的个人或者客户端。

**Fabric网络：**

组成网络的每一个Peer和Orderer，都是得到了批准的、实名的，想要访问这个网络的用户也需要得到批准，并实名签署自己的操作。这就是Fabric网络。

**Fabric实现网络的准入方式：**

* 必须向拥有根证书私钥的机构，申请一个证书。通过这个限制，保证Fabric网络中的参与者的身份都是真实的（参与者的证书被盗用除外）。
* 访问Fabric网络的个人或者客户端，要有一个账号；
* 组成Fabric的网络的每个Peer和Orderer也要有一个账号。
* MSP中存放的就是每个账号的私钥和证书。

**cryptogen命令 和 FabricCA组件：**

* Msp目录可以用Fabric提供的cryptogen命令生成，但是用cryptogen命令生成是非常不灵活的，生产环境中，应当使用Fabric提供的另一个组件FabricCA。
* 注意FabricCA组件：不属于Fabric网络，它是网络之外的一个用来签署证书的服务。借助FabricCA，还可以实现账号的分级，简单说就是一个组织的管理员，可以自由地创建属于该组织的子账号。（这是刚需）

配置：

配置分两种：组件的配置文件、system chain中的配置区块

1组件的配置文件：就是orderer的配置文件orderer.yaml 以及peer的配置文件core.yaml

2 system chain中的配置区块

Configtxgen命令：能将二进制的创世块转换成json格式，生成创世块（system chain的第一个区块）。

configtxgen -inspectBlock ./genesisblock

从Fabric的网络中也可以读取到指定channel的当前配置：

./peer.sh channel fetch config config\_block.pb -c mychannel -o orderer.example.com:7050

只不过读出来的文件都是protobuf格式的，需要在另一个工具configtxlator的帮助下，将其转换为json格式：

configtxlator proto\_decode --input config\_block.pb --type common.Block

更新system chain的过程更繁琐，以在Channel中添加一个新的成员为例。

在Channel中添加一个新的成员，实质就是修改Fabric网络中记录的Channel配置。这个过程需要好几步：

**生成新成员的json格式的配置文件**

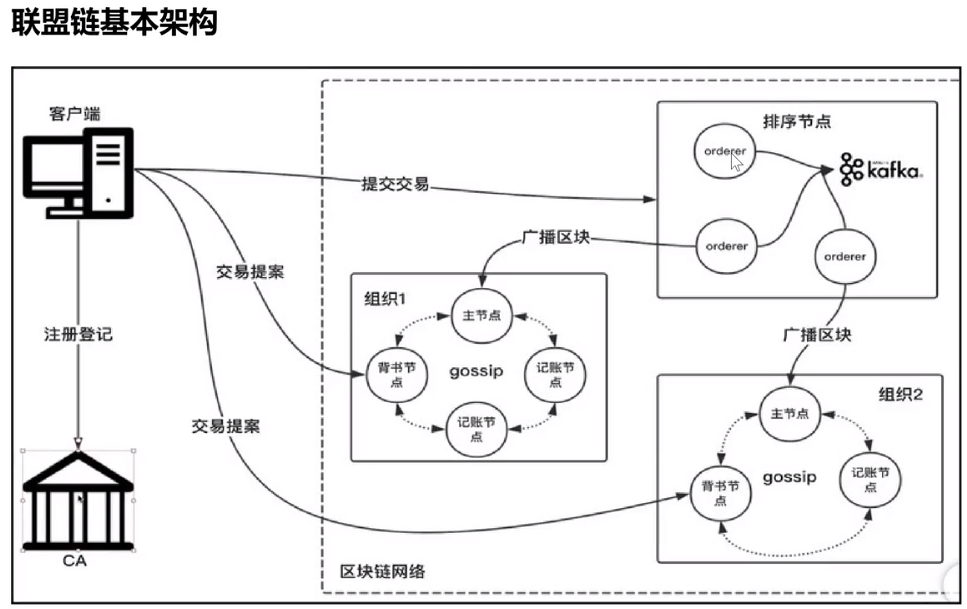
1. 从Fabric中读取Channel最新的配置
2. 将读取的Channel配置转成json格式后，将新成员的json配置加入其中
3. 将修改后的Channel配置和修改前的Channel配置都转成protobuf格式，然后用configtxlator生成更新文件
4. 将更新文件由protobuf格式转换成json格式，加上包含channel信息的信封后，再转换成protobuf格式
5. 将加信封的protobuf格式的更新文件发送给Channel中其它组织的管理员，让他们用自己的私钥进行签署
6. 最后由其中一个管理员将得到足够签署的更新文件提交到网络

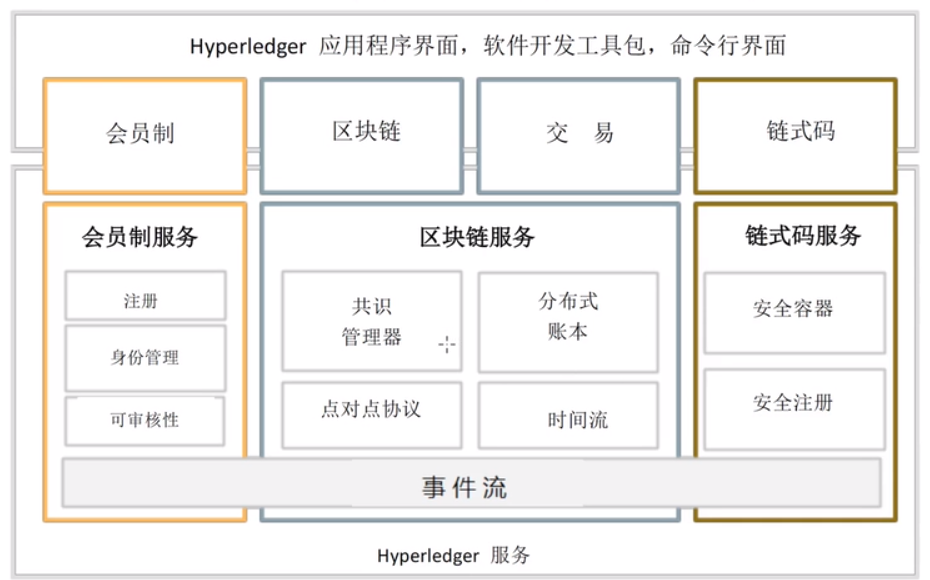
Farbric

* Fabric是一个支持智能合约(smart contracts)的分布式账本(ledger)系统。
* Fabric是私有的只有通过MSP(Membership Service Provider)认证的成员才可以进入到系统，参与合约的缔造与执行。
* Fabric是插件式设计，账本数据有多种存放方式，共识机制(consensus mechanisms)可以切换，支持不同类型的MSP。
* Fabric开发了channel功能，一个channel对应一个账本，只有加入channel的成员可见，可以防止竞争对手知晓交易的细节。
* 账本由两部分组成：全局状态(word state)和交易日志(transaction log）。 全局状态中记录的是当前状态，交易日志中记录了所有的交易记录，全局状态是这些交易记录的执行结果。
* 智能合约(Smart Contracts)用chaincode编写，由区块链外部的应用调用执行，chaincode通常是用来更新账本的。
* Farbric的chaincode目前(2018-02-23 15:08:54)只支持Go语言，以后会支持Java和其它的语言。
* 达成共识的过程中，交易需要严格按照发生的顺序记录到账本中，Farbric提供了多种共识机制(SOLO、Kafka、SBFT…)，建立交易网络的时候根据实际需要选用共识机制。

Fabric的模型主要由以下几个概念组成：

* Assets:交易的资产
* ChainCode:描述交易逻辑的代码
* Ledger Features:账本功能
* Privacy through Channels: channel的私密性，可以对全网开发，也可以只对部分成员开放
  + 包含交易逻辑的ChainCode可以只部署在特定用户端，实现部分公开的效果
  + 还可以在ChainCode中对数据进行加密
* Security & Membership Services: 参与交易的用户都经过认证的可信用户
* Consensus: 交易从发起到被提交到账本的过程中的检验





# Fabric部署