**chaincode：** 链码，Hyperledger Fabric**智能合约**写在链码里并在区块链外部应用程序要和账本发生交易的时候被外部应用程序调用。在大多数情况下，链码只和账本的数据库组件（**世界状态**）交互，而不和交易日志交互。  
**账本**Hyperledger Fabric有一个账本子系统包含两个组件：**世界状态和交易日志**。每一个参与者有一份他们参与的每个Hyperledger Fabric网络的账本的副本。世界状态组件描述了一个给定时间点的账本状态。它是账本的数据库，存储的是**账本当前值**。交易日志组件记录所有导致世界状态当前值的交易。**它是世界状态的更新历史**。这样，账本就是世界状态数据库和交易日志历史的组合体。

世界状态中有一个属性——版本号，版本号从0开始，每当状态更新时版本号就递增。状态更新时会首先检查版本号，以确保当前状态的版本与背书时的版本一致（避免并发更新）。

**区块结构：** 由三个部分组成，分别是区块头、区块数据和区块元数据。  
1.区块头包含三个属性（**区块号**、**当前区块哈希**、**前一个区块的哈希**），当一个区块被创建时写入。

2.区块数据包含的是**排序后的交易列表**。当区块被ordering service创建时写入。

3.区块元数据包括区块的**写入时间，以及区块写入者的证书、公钥和签名。**

**交易：** 在fabric中指的就是对**链代码（即智能合约）的操作**，交易分为两种，**部署交易（deploy transaction）和调用交易（invoke transaction）**。

1. 部署交易  
   部署交易指的是**创建新的链代码**，并且用一个程序作为参数，当一个部署交易成功执行时，链代码就被安装到区块链上了。
2. 调用交易  
   调用交易指的是**运行链代码**，链代码执行时可能会修改相应的状态，并返回输出。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 节点

**节点:** 是区块链的通信实体，是一个逻辑概念，不同类型的多个节点可以运行在同一个物理服务器上。节点主要有以下四种：

**1**.**客户端节点**：客户端必须连接到某一个peer节点或排序服务节点上才能与区块链网络进行通信。**客户端向背书节点**（endorser）**提交交易提案**（transaction proposal），当收集到足够背书后，**向排序服务节点广播交易提案**，进行排序，生成区块。

**2**.**普通节点**peer：peer节点根据所承担的角色又可以分为记账节点（committer）、背书节点（endorser）、主节点（leader）和锚节点（anchor）。

**记账节点**：所有的peer节点都是记账节点（committer），**负责验证排序服务节点区块里的交易，维护状态和总账（Ledger）的副本。**该节点会定期从orderer节点获取包含交易的区块，在对这些区块进行核发验证之后，会把这些区块加入到区块链中。committer节点无法通过配置文件配置，**需要在当前客户端或者命令行发起交易请求的时候手动指定相关的committer节点**。记账节点可以有多个。

**背书节点**：部分节点还会**执行交易并对结果进行签名背书，充当背书节点（endorser）的角色。**背书节点是动态的角色，是与具体链码绑定的。每个链码在实例化的时候都会设置背书策略，指定哪些节点对交易背书后交易才是有效的。并且只有应用程序向它发起交易背书请求的时候才是背书节点，其他时候都是普通的记账节点，只负责验证交易并记账。**背书节点也无法通过配置文件指定，而是由发起交易请求的客户端指定**。背书节点可以有多个。

**锚节点**：peer节点还可以是锚节点（anchor peer），锚节点主要**负责代表组织和其他组织进行信息交换。**每个组织都有一个锚节点，锚节点对于组织来说非常重要，如果锚节点出现问题，当前组织就会与其他组织失去联系。**锚节点的配置信息是在configtxgen模块的配置文件configtx.yaml中配置的**。锚节点只能有一个。

**主节点**：peer节点还可以是主节点（leader peer），能**与排序服务节点通信，负责从排序服务节点获取最新的区块并在组织内部同步。**主节点在整个组织中只能有一个。

**3.排序服务节点orderer：接收包含背书签名的交易，对未打包的交易进行排序生成区块，广播给peer节点。**排序服务提供的是原子广播，保证同一个链上的节点接收到相同的信息，并且有相同的逻辑顺序。

**4.CA节点：**fabric1.0的证书颁发机构，由服务器和客户端组成。**CA节点接收客户端的注册申请，返回注册密码用于用户登录，以便获取身份证书。**区块链上的所有操作都需要验证用户身份。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**org：**fabric系统是通过组织来划分的**，每个组织内都有承担不同功能的peer节点，同时每个组织都有自己对应的fabric-ca服务器，fabric系统中所有的组织共用一个orderer集群。**fabric中的组织在现实世界中可以是一个公司、一个企业，或者一个协会。在fabric中，组织是承担着数据信用责任的区块链系统参与方。  
在设计一个fabric系统时，第一步就是要确定系统的参与方，然后从这些参与者中选出组织（生成对应的组织编号、域名、证书等），然后再确认组织的管理方式。组织的管理方式是指组织在遇到问题时的协作方式（如新组织的加入）。

**channel：**fabric的数据存储结构被设计成多账本体系，每个账本在fabric中被称为channel。每个channel中都有一个完全独立的账本。同一个channel中的所有peer节点都保存一份相同的数据。  
通道由成员（组织）、每个成员的锚节点、账本、链码应用程序和排序服务节点定义。网络上的每个交易都是在一个通道上执行的，在该通道上，每一方都必须经过身份验证和授权才能在该通道上进行交易。加入通道的每一个peer都有其自己的身份，由成员服务提供者（MSP）提供。  
[图片上传失败...(image-75d5dc-1557391569944)]

**MSP：**Membership Service Provider，负责联盟链成员的证书管理，**它定义了哪些RCA以及ICA在链里是可信任的，包括定义了channel上的合作者。**  
每个组织都有自己的证书管理即CA, 及MSP, CA给每个peer颁发证书，MSP授权，赋予相应权限策略。Peer ，applications，end users, administrators orders 必须拥有CA和MSP才能访问链网。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 共识

共识**：**Fabric区块链的网络节点本质上是**互相复制的状态机**，节点之间需要保持相同的账本状态。为了实现分布式节点的一致性，各个节点需要**通过共识过程，对账本状态的变化达成一致性的认同。**  
Fabric区块链的共识过程包括3个阶段：背书、排序和校验。

1.背书

在背书（endorsement）阶段中，背书节点对客户端发来的交易提案进行合法性校验，然后模拟执行链码得到交易结果，最后根据设定的背书逻辑判断是否支持该交易提案。如果背书逻辑决定支持交易提案，会把交易提案签名后发回给客户端。  
客户端通常需要根据链码的背书策略，向一个或者多个成员的背书节点发出背书请求。背书策略会定义需要哪些节点背书交易才有效，例如需要5个成员的背书节点中至少3个同意；或者某个特殊身份的成员支持等。客户端只有在收集足够多的背书节点的交易提案签名，交易才能被视为有效。

2.排序

排序（ordering）阶段就是由排序服务节点对交易进行排序，确定交易之间的时序关系。排序服务把一段时间内收到的交易进行排序，然后把排序后的批量交易打包成数据块（区块），再把区块广播给通道中的成员。采用排序共识方式，各个成员收到的是一组发生顺序相同的交易，**从而保证了所有节点的数据一致性。**  
目前，Hyperledger Fabric有三种交易排序算法：Solo、Kafka、SBFT。  
Solo：只有一个排序服务节点负责接收交易信息并排序，是最简单的一种排序算法，**一般用于开发测试环境中。Solo共识模式属于中心化的处理方式，不支持拜占庭容错。**  
Kafka：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，但不支持拜占庭容错。**  
SBFT：简单拜占庭算法，**支持拜占庭容错的可靠排序算法，包括容忍节点故障以及一定数量的恶意节点。**

排序服务是共识机制中重要的一环，所有交易都要通过排序服务的排序才可以达成全网共识，因此排序服务要避免成为网络上的性能瓶颈。

3.校验

校验（Validation）阶段是节点对排序后的交易进行一系列的检验，包括交易数据的完整性检查、是否重复交易、背书签名是否符合背书策略的要求、交易的读写集是否符合多版本并发控制MVCC（Multiversion Concurrency Control）的校验等。当交易通过了所有校验后，将被标注为合法并写入账本中。因为所有的确认节点都按照相同的顺序检验交易，并且把合法的交易依次写入账本中，因此不同确认节点的状态能够始终保持一致。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 交易流程

**交易流程：**前提假设是各节点已经提前颁发好证书，且已正常启动，并加入已经创建好的通道。此流程介绍的是在已经实例化了的链码通道上从发起一个调用交易到最终结账的全过程。

提交交易提案  
应用程序（客户端节点）构造好**交易提案（交易提案中包含本次交易要调用的合约标识、合约方法和参数信息以及客户端签名等）**请求后，根据背书策略选择背书节点执行交易提案并进行背书签名。背书节点是链代码中背书策略指定的节点。正常情况下背书节点执行后的结果是一致的，只有背书节点对结果的签名不一样。

1. 模拟执行提案并进行背书  
   背书节点在收到交易提案后会进行一些验证，验证通过后，**背书节点会根据当前账本数据模拟执行链码中的业务逻辑并生成读写集（RwSet）。**模拟执行时不会更新账本数据。**然后背书节点对这些读写集进行签名生成提案响应（proposal response）**，然后返回给应用程序。

收集交易的背书（返回模拟执行结果）  
应用程序收到proposal response后会对背书节点的签名进行验证**（所有节点接收到任何消息时都需要先验证消息的合法性）。**如果链码只进行账本查询操作，应用程序只需要检查查询响应，并不会将交易提交给排序服务节点。如果链码对账本进行了invoke操作，则需要提交交易给排序服务进行账本更新（提交前会判断背书策略是否满足）。

1. 构造交易请求并发送给排序服务节点  
   应用程序接收到所有背书节点的签名后，**根据背书签名调用SDK生成交易，并广播给排序服务节点。**其中生成交易的过程很简单，只需要确认所有背书节点的执行结果完全一致，再将交易提案、提案响应和背书签名打包生成交易即可。

排序服务节点对交易进行排序并生成区块  
排序服务节点接收到网络中所有通道发出的交易信息，读取交易信封获取通道名称，按各个通道上交易的接收时间顺序对交易信息进行排序（多通道隔离），生成区块。（在这个过程中，排序服务节点不会关心交易是否正确，只是负责排序和打包。交易的有效性在第7步进行验证）

1. 排序服务节点广播区块给主节点  
   排序服务节点生成区块后会广播给通道上不同组织的主节点。
2. 记账节点验证区块内容并写入到账本  
   所有的peer节点都是记账节点，记录的是节点已加入通道的账本数据。记账节点接收到的排序服务节点生成的区块后，会验证区块交易的有效性，然后提交到本地账本并产生一个生成区块的事件，监听区块事件的应用程序会进行后续的处理。（如果接收的是配置区块，则会更新缓存的配置信息）
3. 主节点在组织内部同步最新的区块  
   如果交易是无效的，也会更新区块，但不会更新世界状态。（区块存储的是操作语句，而世界状态存储的是被处理的数据）

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Fabric其中几个包

十二，msp ✳️  
成员服务提供者（Member Service Provider），提供一组认证相关的密码学机制和协议，用来负责对网络提供证书分发、校验，身份认证管理等。一些成员管理的实现代码等。

十三，orderer ✳️  
在 fabric 1.0 架构中，共识功能被抽取出来，作为单独的 fabric-orderer 模块来实现，**完成核心的排序功能。最核心的功能是实现从客户端过来的 broadcast 请求，和从 orderer 发送到 peer 节点的 deliver 接口。**同时，orderer 需要支持多 channel 的维护。主要包含Solo、kafka及bft三个方法。

十四，peer ✳️  
peer节点的相关主命令模块。作为服务端时候，支持 node 子命令；作为命令行时候，支持 chaincode、channel 等子命令。其中包含一些命令操作的实现等等。

十五，proposals  
一些建议，包含现在对区块的结构优化建议及时序图的呈现。还有其他方面的一些建议文件。

十六，protos ✳️  
Protobuf 格式的数据结构和消息协议都在同一个 protos 包内。  
这里面是所有基本的数据结构（message）定义和 GRPC 的服务（service）接口声明。