HyperLedger Fabric入门指南

区块链是将互相独立的分布式的存储、传输协议、加密机制通过一种特别的方式组合起来，因此这几个相互独立的技术也成就了区块链的三大技术优势：去中心化、共识机制、智能合约，赋予了区块链得以颠覆世界的能力。因为其三大技术优势而受到广泛关注，且目前正处在上升势态。

抛开炒作的代币项目，应用于行业联盟链或直接搭建私链的项目，采用HyperLedger Fabric作为底层平台无疑是最好的选择之一。Hyperledger Project由Linux基金会创办于2015年10月，是一个开源的区块链研发孵化项目，致力于提供可协同开发以区块链为底层的分布式账本。旗下的Fabric项目目标为打造一个提供分布式账本解决方案的平台。本文主要对Fabric的基本概念，网络结构，智能合约等信息介绍以及对未来创建BaaS平台服务的简单构想。

Fabric组成模型

资产定义

资产这里理解为任何具有货币价值的东西，它们都可以通过网络进行交易，无论是有形资产还是无形资产都属于资产。

资产在Hyperledger Fabric中以键-值对集合的形态存在，在通道（channel）中各本地账本可以对其状态提交变更事务。资产可以用二进制或JSON形式表示。

智能合约

链码(chaincode)即Fabric的智能合约，分为系统链码和用户链码。链码的执行由事务排序划分，限制了节点类型间的信任和验证级别，并优化了网络的可伸缩性和性能。

共享账本

在Fabric中产生的所有针对数据状态变更的请求都会生成有序且不可篡改的记录存于账本中。数据状态的变更是由所有参与方认可的智能合约调用事务的结果。每个事务都将产生一组资产键-值对，这些键值对作为创建、更新或删除等操作而同步到所有账本。

账本由区块链（区块根据hash等算法组成的链条）组成，而每一个区块中都存储有一条或一组有序的且不可篡改的记录，也就是一个状态数据库来维护当前的Fabric的状态。每个通道（channel）都有且仅有一个账本，在该通道（channel）中的每个加盟成员的对等点都维护同一份账本。

隐私通道

多通道（channel）交易的设计方案可以确保竞争的企业和受监管的行业在一个公共网络上交换资产时的高度隐私及保密性。

成员服务

Hyperledger Fabric只允许被授权加盟的成员参与数据维护，且成员间相互认可所有的交易都会被彼此发现和跟踪。这种方式提供了一个可信的区块链网络。

共识机制

共识机制的设定是为了达成一致的一种独特的方法，它可以实现企业间所需的灵活性和可伸缩性。

Fabric账本数据结构

要达到数据不可篡改首先从数据结构上来看，也是区块链之所以称之为区块链的原因。如下图1-1所示，每个存储单元包含上一存储单元的hash值（图中hash值的对应关系不完全精确，仅示意用）以及自身存储的交易数据块，可以从表象来看就像把所有数据块连接在一起，称之为“区块链”，形成链状可追述的交易记录。这种链状结构的数据称之为账本数据，保存着所有交易的记录，此外还有一个“世界

状态”，其实质为Key-Value数据库，维护着交易数据的最终状态，便于查询等操作运算，并且每个数据都有其对应的版本号。

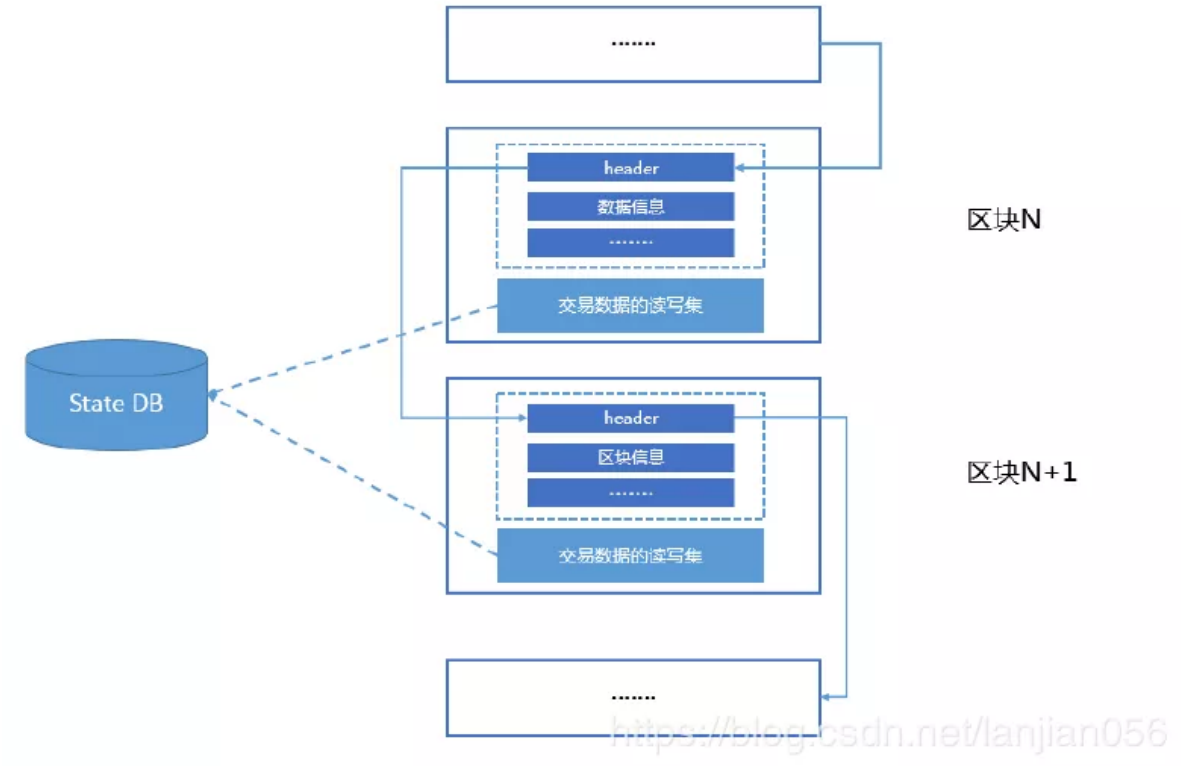

图1-1 区块数据结构图

Fabric涉及的技术及要求

Docker容器

掌握docker的环境搭建。掌握Docker的基本操作命令，包括打包、拆包、tag、容器管理、镜像管理等。掌握Docker Compose和YAML的编排方案，了解如何挂载文件路径、设置端口绑定、确定Network以及掌握环境变量的用法等。

Go语言

掌握go语言的环境部署即可。

gRPC框架

gRPC 是一个高性能、通用的开源RPC框架，其由 Google 公司主要面向移动应用开发并基于HTTP/2 协议标准而设计的，基于 ProtoBuf(Protocol Buffers) 序列化协议开发，且支持众多开发语言。

了解一下即可，与Thrift、Dubbo类似。

CA认证

CA是PKI系统中通信双方信任的实体，被称为可信第三方（Trusted Third Party，简称TTP）。作为可信第三方的行为具有非否认性。

了解即可。

P2P传输

Peer-to-Peer，对等网络，即对等计算机网络，是一种在对等者（Peer）之间分配任务和工作负载的分布式应用架构。

了解即可。

Kafka分布式消息系统

掌握Kafka的环境搭建，了解其在Fabric网络中的担任的作用即可。简单来说，Kafka是一种共识模式，也就是说平等信任（同步复制），所有的HyperLedger Fabric网络加盟方都是可信方，因为消息总是均匀地分布在各处。当Orderer生成新的区块后，通过kafka同步到各Peer节点。各节点将新的区块提交到账本。

Zookeeper

掌握Zookeeper的环境搭建，其作用主要是管理Kafka的broker节点。

SDK（Java/Node.js)

熟练掌握java-sdk的开发工作。

Fabric事务处理流程

1）应用向一个或多个Peer节点发送对事务的背书请求。

2）背书节点执行智能合约，但并不将结果提交到本地账本，只是将结果返回给应用。

3）应用收集所有背书节点的结果后，将结果广播给Orderers集群。（背书策略）

4）Orderers集群执行共识过程，并生成新的区块，通过消息通道批量的将区块发布给Peer 节点。

5）各个Peer节点验证交易，并提交到本地账本中。

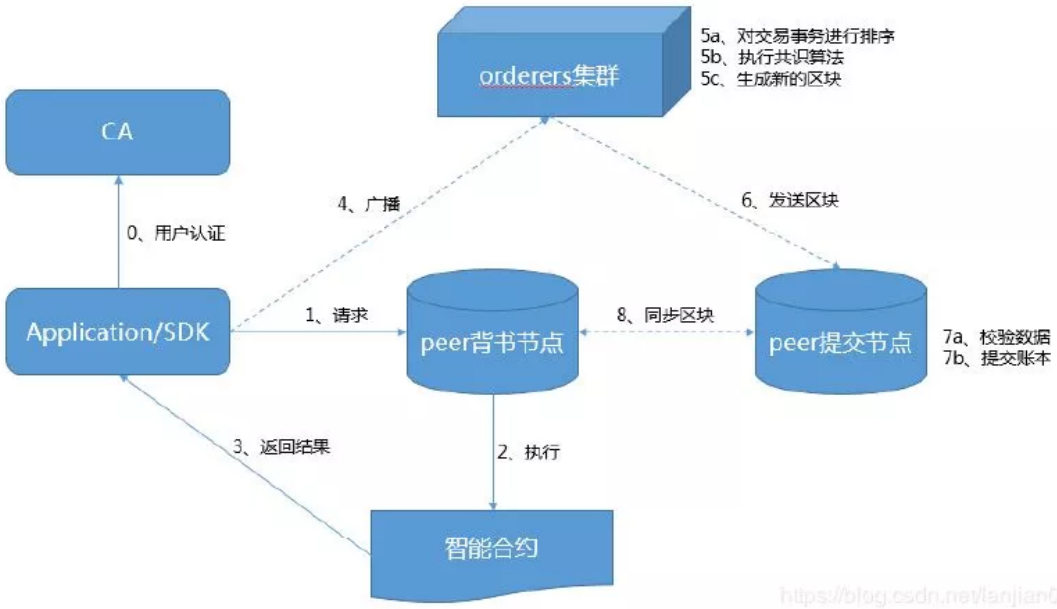

图1-2 事务处理流程图

Fabric网络

构建Fabric网路的基本顺序

1）下载平台对应的二级制文件配置

https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric/hyperledger-fabric/darwin-amd64-1.1.0/

解压后，一般包含的二进制文件有cryptogen,configtxgen,configtxlator

通过configtxgen和cryptogen手动生成证书/密钥以及各项配置文件

2）生成组织证书和秘钥

crypto-config.yaml文件中，主要配置是组织信息、排序节点组织定义、节点组织定义的信息。

命令：./bin/cryptogen generate --config=./crypto-config.yaml

3）定义configtx.yaml配置文件

配置环境变量：export FABRIC_CFG_PATH=configtx.yaml的所在路径

configtx.yaml的五大配置：

Profiles：配置文件。

Organizations：组织节点信息。定义组织ID和MSP的文件位置及组织节点的信息。

Orderer：启动模式。排序服务相关的一些参数配置，包括共识类型、地址码、区块生成的大小、批量处理的数量、超时时间等信息。（很重要的配置，会影响到生产环境，区块生成的速度和区块文件占用的磁盘空间等问题）

Application：应用。（具体参数配置参考官网信息）

Capabilities：功能特性集合。定义Fabirc的网络功能。（具体参数配置参考官网信息）

4）生成orderer源文件

生成genesis.block

命令：

./bin/configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block

5）生成channel源文件

生成channel.tx

命令：./bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/mychannel.tx -channelID mychannel

构建Fabric网路的基本顺序

一个Fabric网络包含一个或多个组织且含有不少于一台排序服务集，一个组织包含一个或多个节点服务，一个节点服务可以加入一个或多个通道，一个通道可以安装一个或多个智能合约。

1）solo多级部署（solo共识）

Solo类型启动多机多节点部署，至少需要两台服务器来完成最小单元的Fabric网络组的工作。具体网络拓扑如下图1-3所示。

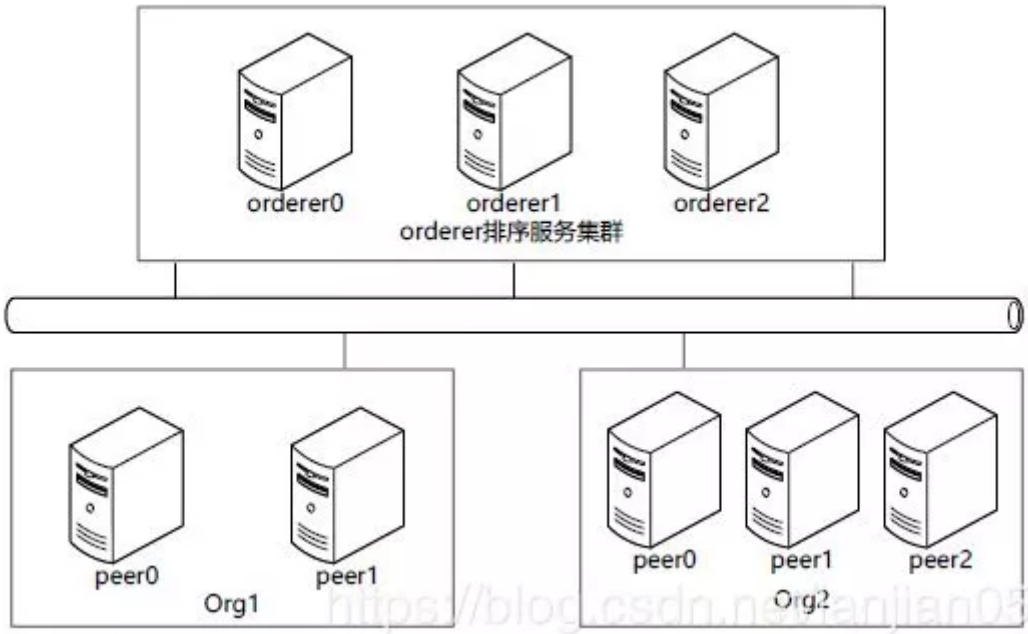

图1-3 solo 共识网络拓扑

2）kafka集群部署（kafka共识）——生产环境必须使用kafka共识

kafka集群的最小单位组成如下：

三个zookeeper节点集群（服务器个数为2n+1个）

四个kafka节点集群

三个orderer排序排序服务器

其他peer节点

具体网络拓扑如下图1-4所示。

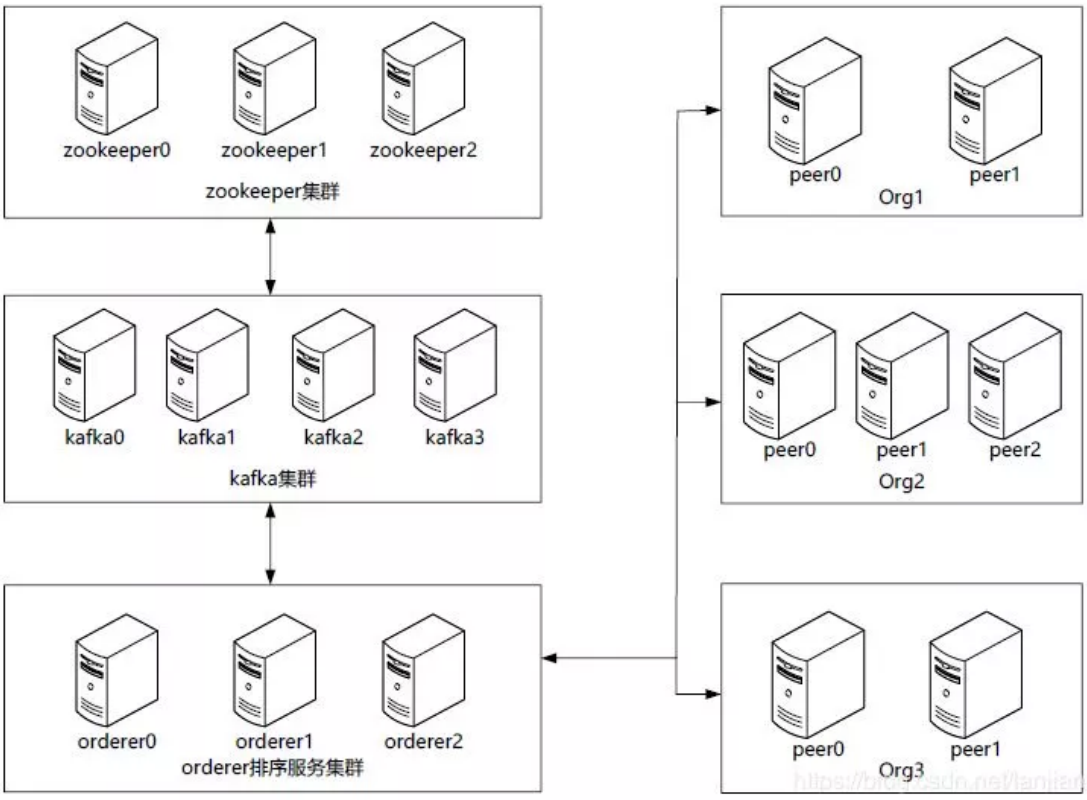

图1-4 kafka共识网络拓扑

什么是智能合约

Fabric的智能合约称为链码（chaincode），分为系统链码和用户链码。系统链码用来实现系统层面的功能，用户链码实现用户的应用功能。智能合约的主要特点如下：

1）由Go语言编写的，最终在Java等其他编程语言中实现了指定的接口。

2）运行在一个背书peer进程独立且安全的Docker容器中。

3）与Fabric网络交互的唯一渠道。

4）生成Transaction的唯一来源（Transactions -> 区块 -> 账本）。

5）智能合约通过应用程序提交事务初始化和管理账本状态。

智能合约与节点的交互流程

1）客户端请求背书节点peer后，节点会通过发送一个链码消息对象（带输入信息，调用者信息）给对应的链码。

2）链码调用ChaincodeBase（基类）里面的invoke方法，通过发送获取数据（getState）和写入数据（putState）消息，向背书节点peer获取账本状态信息和发送预提交状态。

3）链码发送最终输出结果给背书节点peer，节点对输入和输出进行背书签名，完成第一段签名提交。

4）之后客户端收集所有背书节点peer的第一段提交信息，组装事务（transaction）并签名，发送事务到orderer节点进行排序，最终在orderer节点产生区块，并发送到各个peer节点，把输入和输出落到账本上，完成第二段提交过程。

时序图如下图1-5所示。

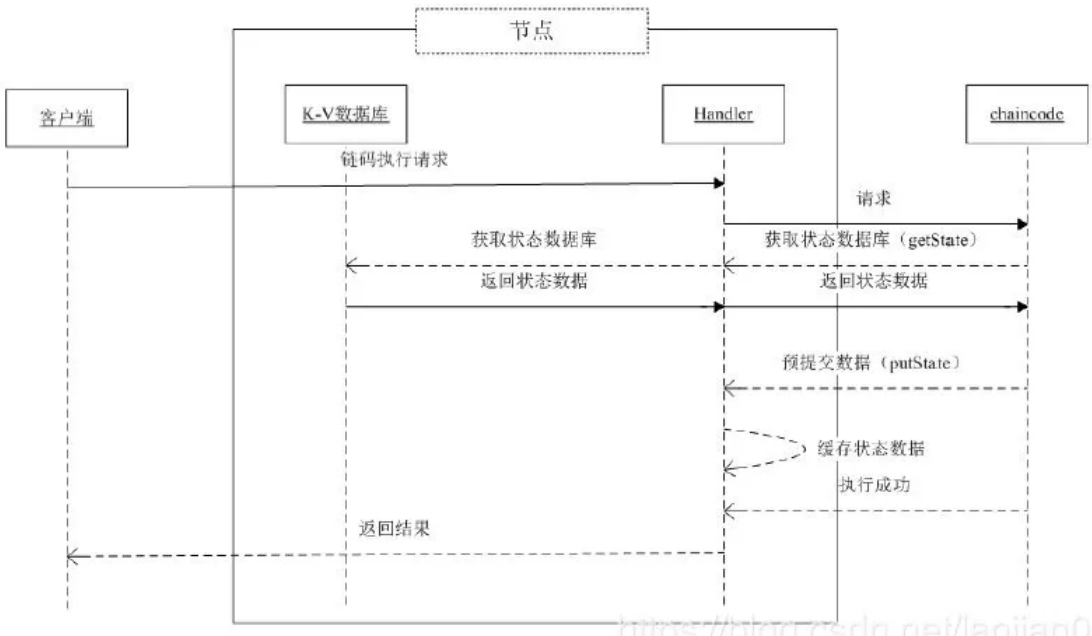

图1-5 链码与节点交互时序图