# BlockChain for Developers

LING Zong,  Ph. D.
Senior Software Engineer / Scientist
IBM Almaden Research Center
San Jose, California, U.S.A.

为开发人员快速启动指南

# IBM Blockchain 101: Quick-start guide for developers

概述IBM区块链平台的功能，然后演示在IBM Cloud中部署IBM区块链平台控制台的过程

An overview of the capabilities of the IBM Blockchain Platform, followed by a demonstration of the process of deploying the IBM Blockchain Platform console in IBM Cloud.

Build a kick-starter blockchain network and start coding with IBM's next-generation Blockchain platform

- This guide shows you how to spin up a blockchain network based on the latest open source Hyperledger Fabric framework using the next generation platform, or building it manually component by component.
- Let’s first review key concepts around developing a business blockchain network.

构建一个快速启动的区块链网络，并使用IBM的下一代区块链平台开始编码

- 本指南向您展示了如何使用下一代平台在最新的开源Hyperledger Fabric框架的基础上构建一个区块链网络，或者手动地逐个组件地构建它。
- 让我们首先回顾一下关于开发商业区块链网络的关键概念。

- Key Concepts
- Developing an Application

# Key Concepts

- Introduction
- Hyperledger Fabric Functionalities
- Hyperledger Fabric Model
- Blockchain network
- Identity
- Membership
- Peers
- Smart Contracts and Chaincode
- Ledger
- The Ordering Service
- Private data
- Channel capabilities
- Use Cases

# Introduction

- Hyperledger Fabric is a platform for distributed ledger solutions underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility, and scalability.

- It is designed to support pluggable implementations of different components and accommodate the complexity and intricacies that exist across the economic ecosystem.

· Hyperledger Fabric是一个分布式账本解决方案的平台，采用模块化架构，提供高度的机密性、弹性、灵活性和可伸缩性。
· 它被设计为支持不同组件的可插拔实现，并适应存在于整个经济生态系统中的复杂性

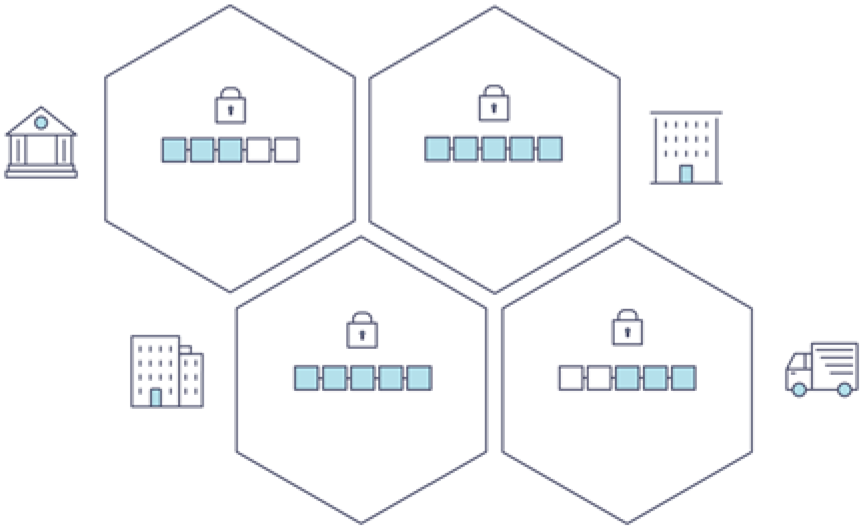
# A Distributed Ledger

- 区块链网络的核心是一个分布式账本，它记录在网络上发生的所有交易。
- 区块链账本通常被描述为分散的，因为它在许多网络参与者之间复制，每个参与者都在维护它。
- 我们将看到，分散化和协作是反映现实世界中企业交换商品和服务方式的强大属性。

- At the heart of a blockchain network is a distributed ledger that records all the transactions that take place on the network.
- A blockchain ledger is often described as **decentralized** because it is replicated across many network participants, each of whom **collaborate** in its maintenance.
- We’ll see that decentralization and collaboration are powerful attributes that mirror the way businesses exchange goods and services in the real world.

- 除了分散和协作之外，记录到区块链中的信息是只能追加的，使用加密技术保证一旦交易被添加到总账中就不能修改。
- 这种不变性的属性使得确定信息的来源变得简单，因为参与者可以确定信息在事件发生后没有被更改。
- 这就是为什么区块链有时被描述为证明系统

- In addition to being decentralized and collaborative, the information recorded to a blockchain is **append-only**, using cryptographic techniques that guarantee that once a transaction has been added to the ledger it cannot be modified.
- This property of “immutability” makes it simple to determine the provenance of information because participants can be sure information has not been changed after the fact.
- It’s why blockchains are sometimes described as **systems of proof**.

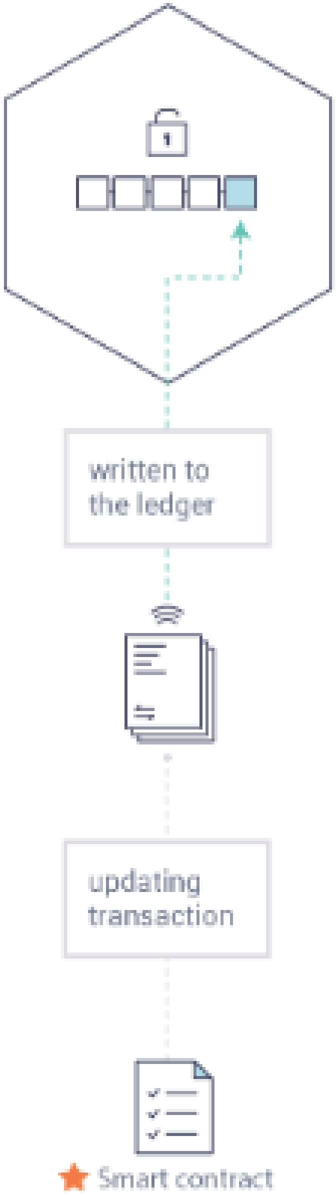


# Smart Contracts

- 为了支持信息的持续更新和启用一套完整的分类账功能(交易、查询等)，区块链网络使用智能合同提供对分类账的控制访问。
- 智能合约不仅是封装信息并使其在网络中保持简单的关键机制，还可以编写为允许参与者自动执行某些方面的交易。

- To support the consistent update of information — and to enable a whole host of ledger functions (transacting, querying, etc) — a blockchain network uses **smart contracts** to provide controlled access to the ledger.
- Smart contracts are not only a key mechanism for encapsulating information and keeping it simple across the network, they can also be written to allow participants to execute certain aspects of transactions **automatically**.
  - A smart contract can, for example, be written to stipulate the cost of shipping an item where the shipping charge changes depending on *how quickly the item arrives*.
  - With the terms agreed to by both parties and written to the ledger, the *appropriate funds change hands automatically* when the item is received.

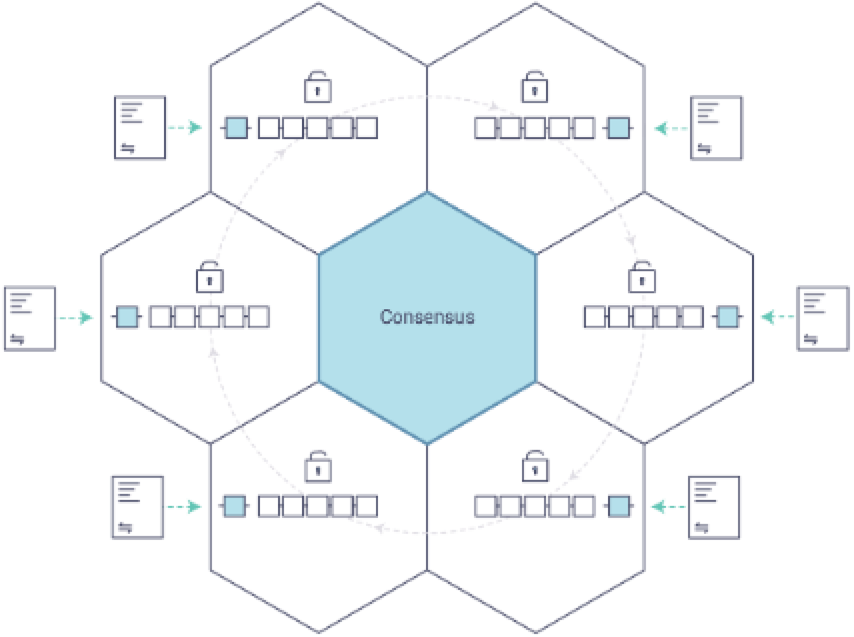
- 例如，智能合同可以规定货物的运输成本，运输费用取决于货物到达的速度。
- 根据双方同意的条款并写入总帐，当收到款项时，相应的资金就会自动转手



# Consensus

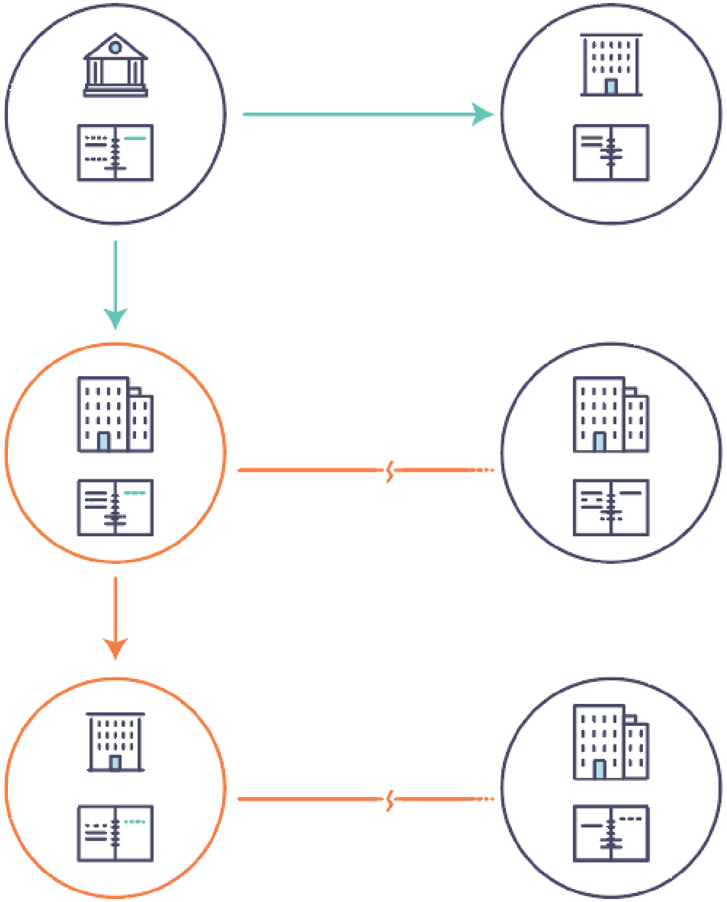
使分类帐交易在网络上保持同步，以确保只有当交易被适当的参与者批准时，分类帐才会更新，当分类帐确实更新时，它们会以同样的顺序更新相同的交易，这一过程称为一致性

The process of keeping the ledger transactions synchronized across the network — to ensure that ledgers update only when transactions are approved by the appropriate participants, and that when ledgers do update, they update with the same transactions in the same order — is called **consensus**.

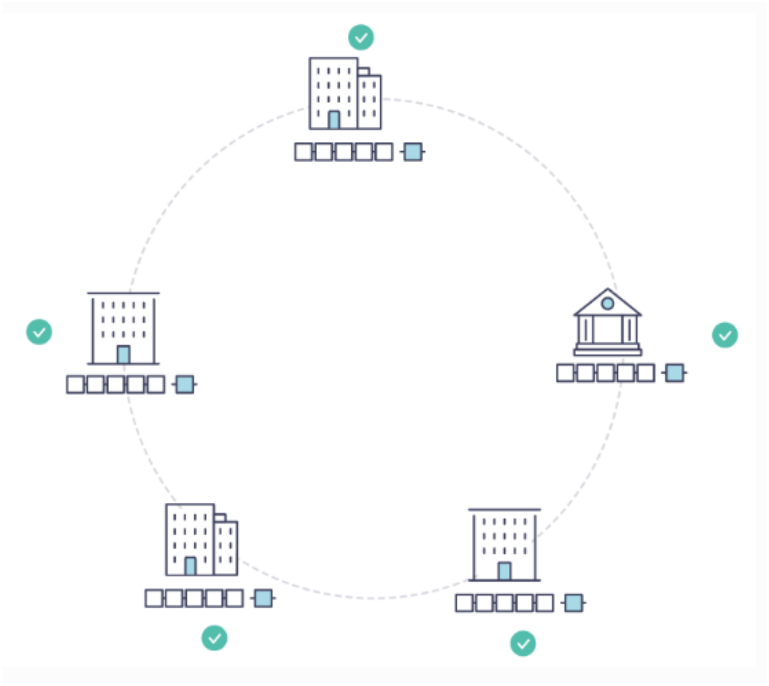


# Why is a Blockchain useful?

Today's Systems of Record



The Blockchain Difference



**sharing of information and processes**

# What is Hyperledger Fabric?

Hyperledger Fabric是Hyperledger的区块链项目之一。和其他区块链技术一样，它有一个分类账，使用智能合同，是一个参与者管理交易的系统

Hyperledger Fabric is one of the blockchain projects within Hyperledger. Like other blockchain technologies, it has a ledger, uses smart contracts, and is a system by which participants manage their transactions.

Hyperledger Fabric与其他一些区块链系统的不同之处在于它是私有的和许可的。

· Hyperledger Fabric网络的成员通过可信的会员服务提供商(MSP)注册，而不是一个允许未知身份参与网络的开放的无许可系统(需要工作证明之类的协议来验证交易和保护网络)

➢Where Hyperledger Fabric breaks from some other blockchain systems is that it is **private** and **permissioned**.

- Rather than an open permissionless system that allows unknown identities to participate in the network (requiring protocols like “proof of work” to validate transactions and secure the network), the members of a Hyperledger Fabric network enroll through a trusted **Membership** **Service Provider** **(MSP)**.

Hyperledger Fabric也提供了几个可插拔的选项。

· 账单数据可以以多种格式存储，共识机制可以交换进出，并支持不同的MSPs

➢Hyperledger Fabric also offers several pluggable options.

- Ledger data can be stored in multiple formats, consensus mechanisms can be swapped in and out, and different MSPs are supported.

Hyperledger Fabric还提供了创建通道的能力，允许一组参与者创建单独的交易账本。

· 对于网络来说，这是一个特别重要的选择，因为有些参与者可能是竞争对手，而不希望他们所做的每一笔交易——例如，他们提供给某些参与者而不是其他参与者的特殊价格——让每个参与者都知道

· 如果两个参与者组成了一个通道，那么这些参与者——没有其他参与者——拥有该通道的分类账的副本

➢Hyperledger Fabric also offers the ability to create **channels**, allowing a group of participants to create a separate ledger of transactions.

- This is an especially important option for networks where some participants might be competitors and not want every transaction they make — a special price they’re offering to some participants and not others, for example — known to every participant.
- If two participants form a channel, then those participants — and no others — have copies of the ledger for that channel.

# Shared Ledger

Hyperledger Fabric有一个由两个组件组成的ledger子系统:世界状态和交易日志。每个参与者都有一份属于他们所属的每个Hyperledger Fabric的账本副本

Hyperledger Fabric has a ledger subsystem comprising two components: the **world state** and the **transaction log**. Each participant has a copy of the ledger to every Hyperledger Fabric network they belong to.

世界状态组件描述了在给定时间点的分类账的状态。这是账单的数据库。交易日志组件记录导致当前世界状态值的所有事务;这是世界状态的更新历史

The **world state component** describes the state of the ledger at a given point in time. It’s the **database of the ledger**. The transaction log component records all transactions which have resulted in the current value of the world state; it’s the update history for the world state.

因此，分类帐是世界状态数据库和交易日志历史的组合。

**The ledger, then, is a combination of the world state database and the transaction log history.**

分类账有一个世界状态的可替换数据存储。默认情况下，这是一个LevelDB键-值存储数据库。交易日志不需要是可插入的。它只是简单地记录区块链网络使用的分类账数据库的前后值

The ledger has a replaceable data store for the world state. By default, this is a LevelDB key-value store database. The transaction log does not need to be pluggable. It simply records the before and after values of the ledger database being used by the blockchain network.

# Smart Contracts

Hyperledger Fabric智能合约是用链码编写的，在区块链外部的应用程序需要与账本交互时调用该合约

Hyperledger Fabric smart contracts are written in **chaincode** and are invoked by an application external to the blockchain when that application needs to interact with the ledger.

在大多数情况下，链码只与分类账的数据库组件、世界状态(例如查询它)交互，而不是与事务日志交互

In most cases, chaincode interacts only with the database component of the ledger, the world state (querying it, for example), and not the transaction log.

链码可以用几种编程语言实现。目前支持Go和Node

Chaincode can be implemented in several programming languages. Currently, Go and Node are supported.

# Consensus

交易必须按照它们发生的顺序写入分类账，即使它们可能在网络内的不同参与者之间。要做到这一点，必须建立交易的顺序，并对错误(或恶意)插入分类账的不良交易进行拒绝

Transactions must be written to the ledger in the order in which they occur, even though they might be between different sets of participants within the network. For this to happen, the order of transactions must be established and a method for rejecting bad transactions that have been inserted into the ledger in error (or maliciously) must be put into place.

这是计算机科学的一个深入研究的领域，有许多方法来实现它，每一种都有不同的权衡。
· 例如，PBFT(实用拜占庭容错)可以为文件副本相互通信提供一种机制，以保持每个副本的一致性，即使在损坏的情况下也是如此。
· 另一种情况是，在比特币中，排序通过一种叫做 “ 挖掘 ” 的过程发生，在这个过程中，相互竞争的计算机竞相解决一个密码难题，这个难题定义了所有进程随后建立的顺序

This is a thoroughly researched area of computer science, and there are many ways to achieve it, each with different trade-offs.

- ◆ For example, PBFT (Practical Byzantine Fault Tolerance) can provide a mechanism for file replicas to communicate with each other to keep each copy consistent, even in the event of corruption.
- ◆ Alternatively, in Bitcoin, ordering happens through a process called mining where competing computers race to solve a cryptographic puzzle which defines the order that all processes subsequently build upon.

Hyperledger Fabric被设计用来允许网络启动者选择一种最能代表参与者之间关系的共识机制。和隐私一样，需求也有不同的范围;从高度结构化的人际关系网络到更加对等的人际关系网络。

Hyperledger Fabric has been designed to allow network starters to choose a consensus mechanism that best represents the relationships that exist between participants. As with privacy, there is a spectrum of needs; from networks that are highly structured in their relationships to those that are more peer-to-peer.

Hyperledger Fabric 共识机制目前包括SOLO，Kafka和Raft

Hyperledger Fabric consensus mechanisms currently include SOLO, Kafka, and Raft.

# Privacy

· 根据网络的需要，B2B网络中的参与者可能对共享多少信息非常敏感。对于其他网络，隐私将不是首要问题。
· Hyperledger Fabric支持隐私(使用通道)是关键运营需求的网络，以及相对开放的网络

- Depending on the needs of a network, participants in a Business-to-Business (B2B) network might be extremely sensitive about how much information they share. For other networks, privacy will not be a top concern.

- Hyperledger Fabric supports networks where privacy (using **channels**) is a key operational requirement as well as networks that are comparatively open.

# More

Identity (conceptual documentation)
A conceptual doc that will take you through the critical role identities play in a Fabric network (using an established PKI structure and x.509 certificates).
概念性文档，它将带您了解身份在Fabric网络中扮演的关键角色(使用已建立的PKI结构和x.509证书)

Membership (conceptual documentation)
Talks through the role of a Membership Service Provider (MSP), which converts identities into roles in a Fabric network.
通过会员服务提供者(MSP)的角色进行对话，MSP将身份转换为组织网络中的角色

Peers (conceptual documentation)
Peers — owned by organizations — host the ledger and smart contracts and make up the physical structure of a Fabric network. 对等方——由组织拥有——托管账本和智能合同，并构成一个Fabric网络的物理结构

Building Your First Network (tutorial)
Learn how to download Fabric binaries and bootstrap your own sample network with a sample script. Then tear down the network and learn how it was constructed one step at a time.
学习如何下载Fabric二进制文件，并使用示例脚本引导您自己的示例网络。然后分解网络，一步一步地了解它是如何构建的。

Writing Your First Application (tutorial)
Deploys a very simple network — even simpler than Build Your First Network — to use with a simple smart contract and application. 部署一个非常简单的网络——甚至比构建您的第一个网络更简单——与一个简单的智能合约和应用程序一起使用

Transaction Flow
A high level look at a sample transaction flow. 从高层次看一个示例交易流

Hyperledger Fabric Model
A high level look at some of components and concepts brought up in this introduction as well as a few others and describes how they work together in a sample transaction flow.
简要介绍了本介绍中提到的一些组件和概念以及其他一些概念，并描述了它们如何在示例交易流中一起工作

# Key Concepts

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

Ledger

The Ordering Service

Private data

Channel capabilities

Use Cases

# Hyperledger Fabric Functionalities

Hyperledger Fabric是分布式账本技术(DLT)的一个实现，在一个模块化的区块链架构中，提供企业级的网络安全、可伸缩性、保密性和性能。Hyperledger Fabric提供以下区块链网络功能

Hyperledger Fabric is an implementation of distributed ledger technology (DLT) that delivers enterprise-ready network security, scalability, confidentiality and performance, in a modular blockchain architecture. Hyperledger Fabric delivers following blockchain network functionalities:

## Identity management

身份管理

为了启用授权网络，Hyperledger Fabric提供了一个会员身份服务来管理用户id和认证网络上的所有参与者。访问控制列表可用于通过特定网络操作的授权提供额外的权限层。例如，特定的用户ID可以被允许调用链码应用程序，但不能部署新的链码。

To enable permissioned networks, Hyperledger Fabric provides a membership identity service that manages user IDs and authenticates all participants on the network. Access control lists can be used to provide additional layers of permission through authorization of specific network operations. For example, a specific user ID could be permitted to invoke a chaincode application, but be blocked from deploying new chaincode.

## Privacy and confidentiality

隐私和保密

Hyperledger Fabric使竞争的商业利益，和任何要求私人、机密的交易的团体，在同一许可的网络上共存。私有通道是受限制的消息传递路径，可用于为网络成员的特定子集提供事务隐私和机密性。通道上的所有数据，包括交易、成员和通道信息，对于没有显式授予访问该通道权限的任何网络成员都是不可见的和不可访问的

Hyperledger Fabric enables competing business interests, and any groups that require private, confidential transactions, to coexist on the same permissioned network. Private **channels** are restricted messaging paths that can be used to provide transaction privacy and confidentiality for specific subsets of network members. All data, including transaction, member and channel information, on a channel are invisible and inaccessible to any network members not explicitly granted access to that channel
.

## Efficient processing

有效的处理

Hyperledger Fabric按节点类型分配网络角色。为了向网络提供并发性和并行性，事务的执行与事务的排序和提交是分离的。在对事务排序之前执行事务使每个对等节点能够同时处理多个事务。这种并发执行提高了每个对等点上的处理效率，并加快了向订购服务交付事务的速度。除了支持并行处理之外，劳动分工为排序节点减轻了事务执行和账本维护的负担，而对等节点则从排序(一致)工作负载中解脱出来。角色的这种分歧还限制了授权和身份验证所需的处理;所有对等节点不必信任所有有序节点，反之亦然，因此一个节点上的进程可以独立于另一个节点的验证运行。

Hyperledger Fabric assigns network roles by node type. To provide concurrency and parallelism to the network, transaction execution is separated from transaction ordering and commitment. Executing transactions prior to ordering them enables each peer node to process multiple transactions simultaneously. This concurrent execution increases processing efficiency on each peer and accelerates delivery of transactions to the ordering service.

In addition to enabling parallel processing, the division of labor unburdens ordering nodes from the demands of transaction execution and ledger maintenance, while peer nodes are freed from ordering (consensus) workloads. This bifurcation of roles also limits the processing required for authorization and authentication; all peer nodes do not have to trust all ordering nodes, and vice versa, so processes on one can run independently of verification by the other.

## Chaincode functionality

链码功能

链码应用程序对通道上特定类型的交易调用的逻辑进行编码。例如，为资产所有权变更定义参数的链码确保所有转移所有权的交易都遵守相同的规则和要求。系统链码是定义整个信道运行参数的链码。生命周期和配置系统链码定义了通道的规则;背书和验证系统链码定义了背书和验证交易的要求。

Chaincode applications encode logic that is invoked by specific types of transactions on the channel. Chaincode that defines parameters for a change of asset ownership, for example, ensures that all transactions that transfer ownership are subject to the same rules and requirements. **System chaincode** is distinguished as chaincode that defines operating parameters for the entire channel. Lifecycle and configuration system chaincode defines the rules for the channel; endorsement and validation system chaincode defines the requirements for endorsing and validating transactions.

## Modular design

模块化设计

Hyperledger Fabric实现了一个模块化的架构，为网络设计人员提供了功能选择。例如，身份、排序(一致意见)和加密等特定算法可以插入任何Hyperledger Fabric网络。其结果是任何行业或公共领域都可以采用的通用区块链架构，并保证其网络可以跨市场、监管和地理边界进行互操作。

Hyperledger Fabric implements a modular architecture to provide functional choice to network designers. Specific algorithms for identity, ordering (consensus) and encryption, for example, can be plugged in to any Hyperledger Fabric network. The result is a universal blockchain architecture that any industry or public domain can adopt, with the assurance that its networks will be interoperable across market, regulatory and geographic boundaries.

# Key Concepts

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

Ledger

The Ordering Service

Private data

Channel capabilities

Use Cases

# Hyperledger Fabric Model

本节概述了Hyperledger Fabric中编织的关键设计特性，以实现其全面的、可定制的、企业级区块链解决方案的承诺

This section outlines the key design features woven into Hyperledger Fabric that fulfill its promise of a comprehensive, yet customizable, enterprise blockchain solution:

•Assets — Asset definitions enable the exchange of almost anything with monetary value over the network, from whole foods to antique cars to currency futures. 资产定义使几乎所有具有货币价值的东西都可以通过网络进行交换，从全食超市(whole foods)到古董车，再到货币期货

链码执行从事务排序中划分，限制了跨节点类型所需的信任和验证级别，并优化了网络的可伸缩性和性能

•Chaincode — Chaincode execution is partitioned from transaction ordering, limiting the required levels of trust and verification across node types, and optimizing network scalability and performance.

不可变的共享账本为每个通道编码整个交易历史，并包含类似sql的查询功能，用于有效的审计和争议解决

•Ledger Features — The immutable, shared ledger encodes the entire transaction history for each channel, and includes SQL-like query capability for efficient auditing and dispute resolution.

通道和私有数据收集使私有和机密的多边交易成为可能，这通常是在公共网络上交换资产的竞争企业和受管制行业所要求的。

•Privacy — Channels and private data collections enable private and confidential multi-lateral transactions that are usually required by competing businesses and regulated industries that exchange assets on a common network.

许可的会员资格提供了一个可信的区块链网络，在这个网络中，参与者知道所有的交易都可以被授权的监管机构和审计人员检测和跟踪

•Security & Membership Services — Permissioned membership provides a trusted blockchain network, where participants know that all transactions can be detected and traced by authorized regulators and auditors.

达成共识的独特方法使企业能够实现所需的灵活性和可伸缩性

•Consensus — A unique approach to consensus enables the flexibility and scalability needed for the enterprise.

# Assets

- 资产可以是有形的(房地产和硬件)，也可以是无形的(合同和知识产权)。
- Hyperledger Fabric提供了使用链码交易修改资产的能力。
- 资产在Hyperledger Fabric以键-值对的集合表示,与状态改变交易记录一起作为通道分类帐。资产可以用二进制和/或JSON形式表示

- Assets can range from the tangible (real estate and hardware) to the intangible (contracts and intellectual property).

- Hyperledger Fabric provides the ability to modify assets using chaincode transactions.

- Assets are represented in Hyperledger Fabric as a collection of key-value pairs, with state changes recorded as transactions on a Channel ledger. Assets can be represented in binary and/or JSON form.

# Chaincode

- 链码是定义资产的软件，以及用于修改资产的交易指令;换句话说，这是业务逻辑。
- 链码执行读取或更改键值对或其他状态数据库信息的规则。
- 链码函数根据分类账的当前状态数据库执行，并通过一个交易提议启动。
- 链码执行产生一组键-值写(写集)，可以提交给网络并应用到所有对等点上的账单上

- ➢ Chaincode is software defining an asset or assets, and the transaction instructions for modifying the asset(s); in other words, it’s the **business logic**.
- ➢ Chaincode enforces the rules for reading or altering key-value pairs or other state database information.
- ➢ Chaincode functions execute against the ledger’s current state database and are initiated through a transaction proposal.
- ➢ Chaincode execution results in a set of key-value writes (write set) that can be submitted to the network and applied to the ledger on all peers.

# Ledger Features

分类账是fabric中所有状态转换的有序的、抗篡改的记录。
· 状态转换是参与方提交的链代码调用(交易)的结果。
· 每个交易产生一组资产键值对，这些键值对在创建、更新或删除时提交给分类账。

The ledger is the sequenced, tamper-resistant **record** of all state transitions in the fabric.

- State transitions are a result of chaincode invocations (‘transactions’) submitted by participating parties.
- Each transaction results in a set of asset key-value pairs that are committed to the ledger as creates, updates, or deletes.

分类账由一个区块链(链)组成，以块的形式存储不可变的、顺序的记录，以及一个状态数据库来维护当前的结构状态。每个通道有一个分类账。每个对等点为它们所属的每个通道维护一个分类账副本

The ledger is comprised of a blockchain (‘chain’) to store the immutable, sequenced record in blocks, as well as a state database to maintain current fabric state. There is **one ledger per channel**. Each peer maintains a copy of the ledger for each channel of which they are a member.  Some features of a Fabric ledger:

- 使用基于键的查找、范围查询和复合键查询的查询和更新分类帐
  Query and update ledger using key-based lookups, range queries, and composite key queries
- 使用富查询语言的只读查询(如果使用CouchDB作为状态数据库)
  Read-only queries using a rich query language (if using CouchDB as state database)
- 只读历史查询——查询一个键的分类账历史，支持数据来源场景
  Read-only history queries — Query ledger history for a key, enabling data provenance scenarios
- 交易由链码(读集)中读取的键/值和链码(写集)中写入的键/值的版本组成。
  Transactions consist of the versions of keys/values that were read in chaincode (read set) and keys/values that were written in chaincode (write set)
- 交易包含每个认可的对等点的签名，并提交给订购服务
  Transactions contain signatures of every endorsing peer and are submitted to ordering service
- 事务被排序成块，并从一个订购服务交付给通道上的对等点
  Transactions are ordered into blocks and are “delivered” from an ordering service to peers on a channel
- 对等点根据背书策略验证事务并执行策略
  Peers validate transactions against endorsement policies and enforce the policies
- 在添加块之前，将执行版本控制检查，以确保所读取的资产的状态自链码执行以来没有发生更改
  Prior to appending a block, a versioning check is performed to ensure that states for assets that were read have not changed since chaincode execution time
- 一旦验证并提交了事务，就具有不可更改性
  There is immutability once a transaction is validated and committed
- 通道分类账包含定义策略、访问控制列表和其他相关信息的配置块
  A channel’s ledger contains a configuration block defining policies, access control lists, and other pertinent information
- 通道包含会员服务提供者实例，允许从不同的证书颁发机构派生加密材料
  Channels contain Membership Service Provider instances allowing for crypto materials to be derived from different certificate authorities

# Privacy

Hyperledger Fabric在每个通道的基础上使用了一个不可变的账本，以及可以操作和修改资产当前状态的链码(即更新键值对)。一个账本存在于一个通道的范围内，它可以在整个网络中被共享(假设每个参与者都在一个公共通道上操作)，或者它可以被私有化，只包括一组特定的参与者。

Hyperledger Fabric employs an immutable ledger on a per-channel basis, as well as chaincode that can manipulate and modify the current state of assets (i.e. update key-value pairs). A ledger exists in the scope of a channel — it can be shared across the entire network (assuming every participant is operating on one common channel) — or it can be privatized to include only a specific set of participants.

为了解决那些想要在完全透明和隐私之间架起桥梁的场景,链码只能安装在需要访问资产状态以执行读写操作的对等点上(换句话说，如果链码没有安装在对等点上，它将无法与分类账正确交互)

- In order to solve scenarios that want to bridge the gap between total **transparency and privacy**, chaincode can be installed only on peers that need to access the asset states to perform reads and writes (in other words, if a chaincode is not installed on a peer, it will not be able to properly interface with the ledger).

当该通道上的组织子集需要保持其交易数据的机密性时，将使用私有数据集合(collection)在私有数据库中隔离该数据，从逻辑上与通道分类账分离，只有经过授权的组织子集可以访问该数据。因此，通道对更广泛的网络保持交易私有，而集合在通道上的组织子集之间保持数据私有

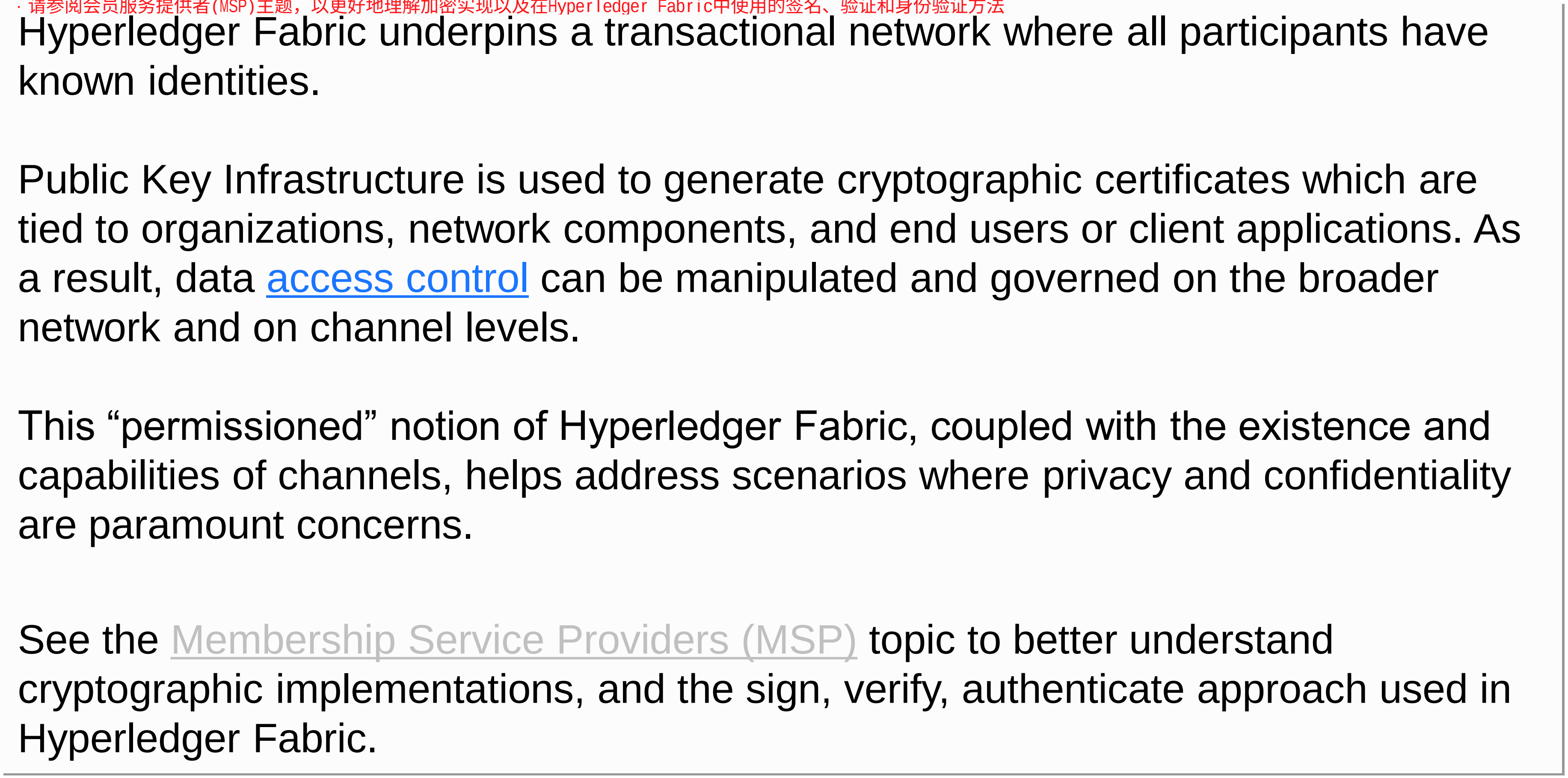
- When a subset of organizations on that channel need to keep their transaction data confidential, a **private data collection** (collection) is used to segregate this data in a private database, logically separate from the **channel ledger**, accessible only to the authorized subset of organizations. Thus, channels keep transactions private from the broader network whereas collections keep data private between subsets of organizations on the channel.

为了进一步混淆数据，在将交易发送到排序服务和向分类账附加块之前，可以使用常用的加密算法(部分或全部)对链码中的值进行加密(如AES)。一旦加密的数据被写入账单，只有拥有相应密钥的用户才能对其进行解密，该密钥用于生成密码文本

- To further obfuscate the data, values within chaincode can be encrypted (in part or in total) using common cryptographic algorithms such as AES before sending transactions to the ordering service and appending blocks to the ledger. Once encrypted data has been written to the ledger, it can be **decrypted only by a user** in possession of the corresponding key that was used to generate the cipher text.

# Security & Membership Services

- Hyperledger Fabric支撑着交易网络，其中所有参与者都拥有已知身份。
- 公钥基础设施用于生成绑定到组织、网络组件和最终用户或客户端应用程序的加密证书。因此，可以在更广泛的网络和通道级别上操纵和管理数据访问控制。
- 这种Hyperledger Fabric的许可概念，加上通道的存在和能力，有助于解决隐私和机密性中最重要的问题。
- 请参阅会员服务提供者(MSP)主题，以更好地理解加密实现以及在Hyperledger Fabric中使用的签名、验证和身份验证方法



Hyperledger Fabric underpins a transactional network where all participants have known identities.

Public Key Infrastructure is used to generate cryptographic certificates which are tied to organizations, network components, and end users or client applications. As a result, data access control can be manipulated and governed on the broader network and on channel levels.

This “permissioned” notion of Hyperledger Fabric, coupled with the existence and capabilities of channels, helps address scenarios where privacy and confidentiality are paramount concerns.

See the Membership Service Providers (MSP) topic to better understand cryptographic implementations, and the sign, verify, authenticate approach used in Hyperledger Fabric.

# Consensus

In distributed ledger technology, consensus has recently become synonymous with a specific algorithm, within a single function.

在分布式账本技术中，共识最近已成为单一功能内的一种特定算法的同义词。
· 然而，共识不只是简单地同意交易的顺序，这种区别在Hyperledger Fabric中通过其在整个交易流程中的基本作用得到强调，从提议和认可，到排序、确认和提交。
· 简而言之，共识被定义为对包含一个块的一组交易的正确性进行完整的验证

- However, consensus encompasses more than simply agreeing upon the order of transactions, and this differentiation is highlighted in Hyperledger Fabric through its fundamental role in the **entire transaction flow**, from proposal and endorsement, to ordering, validation and commitment.
- In a nutshell, consensus is defined as **full-circle verification of the correctness** of a set of transactions comprising a block.

当区块的交易的顺序和结果满足显式的策略标准检查时，最终就会达成一致。
· 这些检查和平衡发生在交易的生命周期中，并包括使用背书策略来规定哪些特定成员必须背书某个事务类，以及使用系统链码来确保这些策略得到实施和维护。

Consensus is achieved ultimately when the order and results of a block’s transactions have met the **explicit policy criteria checks**.

· 在提交之前，对等方将使用这些系统链码，以确保有足够的背书，并确保它们来自适当的实体。
· 此外，在任何包含事务的块被附加到分类账之前，在对分类账的当前状态达成一致或同意时，将进行版本控制检查。
· 这最后一项检查提供了对重复消费操作和其他可能危及数据完整性的威胁的保护，并允许针对非静态变量执行函数

- These checks and balances take place during the lifecycle of a transaction, and include the usage of endorsement policies to dictate which specific members must endorse a certain transaction class, as well as system chaincodes to ensure that these **policies are enforced** and upheld.
- Prior to commitment, the peers will employ these system chaincodes to make sure that **enough endorsements** are present, and that they were derived from the appropriate entities.
- Moreover, a **versioning check** will take place during which the current state of the ledger is agreed or consented upon, before any blocks containing transactions are appended to the ledger.
- This final check provides protection **against double spend** operations and other threats that might compromise data integrity, and allows for functions to be executed against non-static variables.

除了需要进行大量的背书、有效性和版本控制检查之外，还需要在交易流的各个方向进行身份验证。
· 访问控制列表是在网络的分层层(从排序服务到信道)上实现的，当交易提议通过不同的体系结构组件时，有效负载会重复签名、验证和验证

In addition to the multitude of endorsement, validity and versioning checks that take place, there are also ongoing **identity verifications** happening in all directions of the transaction flow.

- Access control lists are implemented on hierarchical layers of the network (ordering service down to channels), and payloads are repeatedly signed, verified and authenticated as a transaction proposal passes through the different architectural components.

最后，共识不仅限于对一批交易的商定订单;相反，它是作为交易从提议到提交过程中正在进行的验证的副产品而实现的一个总体特征

To conclude, consensus is not merely limited to the agreed upon order of a batch of transactions; rather, it is an overarching characterization that is achieved as a byproduct of the ongoing verifications that take place during a transaction’s journey **from proposal to commitment**.

# Key Concepts

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

Ledger

The Ordering Service

Private data

Channel capabilities

Use Cases

# Blockchain network

区块链网络是一种技术基础设施，为应用程序提供账本和智能合同(链码)服务

A blockchain network is a **technical infrastructure** that provides ledger and smart contract (chaincode) services to applications.

首先，智能合同用于生成交易，这些交易随后被分发到网络中的每个对等节点，在那里它们被永久地记录在它们的账本副本中。应用程序的用户可能是使用客户机应用程序的终端用户或区块链网络管理员

Primarily, smart contracts are used to generate transactions which are subsequently distributed to every peer node in the network where they are immutably recorded on their copy of the ledger. The users of applications might be end users using client applications or blockchain network administrators.

Ledger. One per channel. Comprised of the Blockchain and the World state
Smart contract (aka chaincode)
Peer nodes
Ordering service
Channel
Certificate Authority

在大多数情况下，多个组织组成一个联盟来组成网络，它们的权限由联盟在最初配置网络时同意的一组策略决定

In most cases, multiple organizations come together as a **consortium** to form the network and their permissions are determined by a set of policies that are agreed by the consortium when the network is originally configured.

此外，网络策略可以随着时间的推移而改变，这取决于联盟中组织的一致意见，我们将在讨论修改策略的概念时发现这一点

Moreover, network policies can change over time subject to the agreement of the organizations in the consortium, as we'll discover when we discuss the concept of *modification policy*.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/network/network.html

# Key Concepts

Introduction
Hyperledger Fabric Functionalities
Hyperledger Fabric Model
Blockchain network
Identity
Membership
Peers
Smart Contracts and Chaincode
Ledger
The Ordering Service
Private data
Channel capabilities
Use Cases

# Identity

区块链网络中的不同参与者包括对等点、订货方、客户端应用程序、管理员等等。
·这些参与者(网络内部或外部能够使用服务的活动元素)都有一个封装在X.509数字证书中的数字身份
·这些身份确实很重要，因为它们决定了对资源的确切权限和对区块链网络中参与者拥有的信息的访问

The different **actors** in a blockchain network include peers, orderers, client applications, administrators and more.

- Each of these actors — active elements inside or outside a network able to consume services — has a digital identity encapsulated in an X.509 digital certificate.
- These identities really matter because they **determine the exact permissions over resources and access to information that actors have in a blockchain network.**

此外，数字标识还有一些附加属性，Fabric使用这些属性来确定权限，并且它为标识和相关属性的联合提供一个特殊的名称--主体。
·主体就像userid或groupid一样，但是更加灵活，因为它们可以包含参与者身份的广泛属性，比如参与者的组织、组织单元、角色甚至是参与者的特定身份。
·当我们谈到主体时，它们是决定其权限的属性

A **digital identity** furthermore has some additional attributes that Fabric uses to determine permissions, and it gives the union of an identity and the associated attributes a special name — **principal**.

- Principals are just like userIDs or groupIDs, but a little more flexible because they can include a wide range of properties of an actor's identity, such as the actor's organization, organizational unit, role or even the actor's specific identity.
- When we talk about principals, they are the properties which determine their permissions.

要想验证身份，它必须来自可信的权威机构。
·成员资格服务提供者(MSP)是在Fabric中实现这一点的方式。
·更具体地说，MSP是一个组件，它定义管理组织有效身份的规则。
·Fabric中默认的MSP实现使用X.509证书作为身份，采用了传统的公钥基础设施(PKI)层次模型(稍后详细介绍PKI)。

For an identity to be **verifiable**, it must come from a **trusted** authority.

- A membership service provider (MSP) is how this is achieved in Fabric.
- More specifically, an MSP is a component that defines the rules that govern the valid identities for this organization.
- The default MSP implementation in Fabric uses X.509 certificates as identities, adopting a traditional Public Key Infrastructure (PKI) hierarchical model (more on PKI later).

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html

# Key Concepts

- Introduction
- Hyperledger Fabric Functionalities
- Hyperledger Fabric Model
- Blockchain network
- Identity
- Membership
- Peers
- Smart Contracts and Chaincode
- Ledger
- The Ordering Service
- Private data
- Channel capabilities
- Use Cases

# Membership

会员服务提供者(MSP)——它标识所信任的根CAs和中间CAs定义信任域的成员,例如,一个组织,要么通过列出其成员的身份,或通过识别哪些CAs授权发行有效身份的成员,或,通常会通过两者的结合

· MSP的功能不仅仅是简单地列出谁是网络参与者或通道的成员。MSP可以确定参与者在MSP所代表的组织范围内可能扮演的特定角色(如管理员,或作为子组织的成员),并为在网络和通道上下文中定义访问特权设置基础(如频道管理员、读者、作家)。

· MSP的配置被通知给相应组织成员参与的所有通道(以通道MSP的形式)。

· 除了通道MSP之外，对等点、订货程序和客户端还维护一个本地MSP，以在通道上下文之外对成员消息进行身份验证，并定义特定组件(例如，能够在对等点上安装链码的组件)的权限。

· 此外，MSP还允许对已撤销的标识列表进行标识。

**Membership Service Provider** (MSP) — it identifies which Root CAs and Intermediate CAs are trusted to define the members of a trust domain, e.g., an organization, either by listing the identities of their members, or by identifying which CAs are authorized to issue valid identities for their members, or — as will usually be the case — through a combination of both.

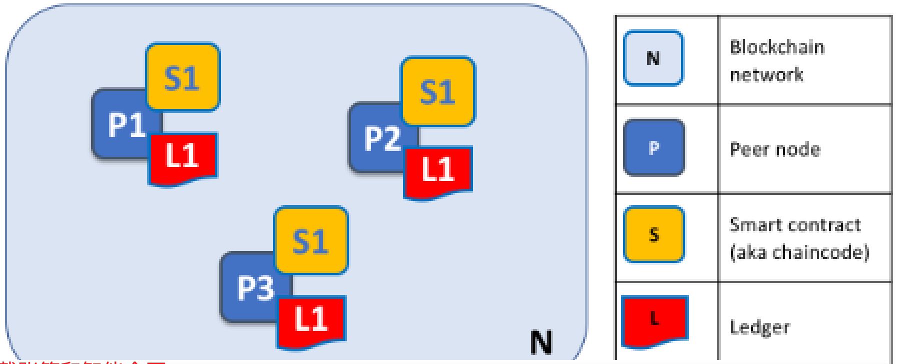
- The power of an MSP goes beyond simply listing who is a network participant or member of a channel.  An MSP can identify specific **roles** an actor might play either within the scope of the organization the MSP represents (e.g., admins, or as members of a sub-organization group), and sets the basis for defining **access privileges** in the context of a network and channel (e.g., channel admins, readers, writers).
- The configuration of an MSP is advertised to all the channels where members of the corresponding organization participate (in the form of a **channel MSP**).
- In addition to the channel MSP, peers, orderers, and clients also maintain a **local MSP** to authenticate member messages outside the context of a channel and to define the permissions over a particular component (who has the ability to install chaincode on a peer, for example).
- In addition, an MSP can allow for the identification of a list of identities that have been revoked.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/membership/membership.html

# Key Concepts

Introduction
Hyperledger Fabric Functionalities
Hyperledger Fabric Model
Blockchain network
Identity
Membership
Peers
Smart Contracts and Chaincode
Ledger
The Ordering Service
Private data
Channel capabilities
Use Cases

# Peers



区块链网络主要由一组对等节点(或简单地说，对等节点)组成。对等点是网络的基本元素，因为它们承载账簿和智能合同

A blockchain network is comprised primarily of a set of *peer nodes* (or, simply, *peers*). Peers are a **fundamental element** of the network because they host ledgers and smart contracts.

回想一下，一个分类帐会永久地记录由智能合约(在Hyperledger Fabric中，它们包含在一个链码中，稍后将对此进行详细介绍)产生的所有交易。智能合同和账簿分别用于封装网络中的共享流程和共享信息

Recall that a ledger immutably records all the transactions generated by smart contracts (which in Hyperledger Fabric are contained in a ***chaincode***, more on this later). Smart contracts and ledgers are used to encapsulate the shared *processes* and shared *information* in a network, respectively.

对等点的这些方面使它们成为理解结构网络的良好起点。

These aspects of a peer make them a good starting point to understand a Fabric network.

区块链网络的其他元素当然是重要的:账簿和智能合同、订货人、策略、通道、应用程序、组织、身份和成员资格，您可以在它们各自的专用章节中阅读更多关于它们的内容

Other elements of the blockchain network are of course important: ledgers and smart contracts, orderers, policies, channels, applications, organizations, identities, and membership, and you can read more about them in their own dedicated sections.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/peers/peers.html

# Key Concepts

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

Ledger

The Ordering Service

Private data

Channel capabilities

Use Cases

# Smart Contracts and Chaincode

从应用程序开发人员的角度来看，智能合同和账本构成了Hyperledger Fabric区块链系统的核心

- From an application developer’s perspective, a **smart contract**, together with the ledger, form the heart of a Hyperledger Fabric blockchain system.

分类账保存一组业务对象的当前和历史状态的事实，而智能合约定义可执行逻辑，该逻辑生成添加到分类账的新事实

- Whereas a ledger holds facts about the current and historical state of a set of business objects, a smart contract defines the **executable logic** that generates new facts that are added to the ledger.

链码通常由管理员用于对相关的智能合约进行分组，以便进行部署，但也可以用于Fabric的低级系统编程

- A **chaincode** is typically used by administrators to group related smart contracts for deployment, but can also be used for low level system programming of Fabric.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/smartcontract/smartcontract.html

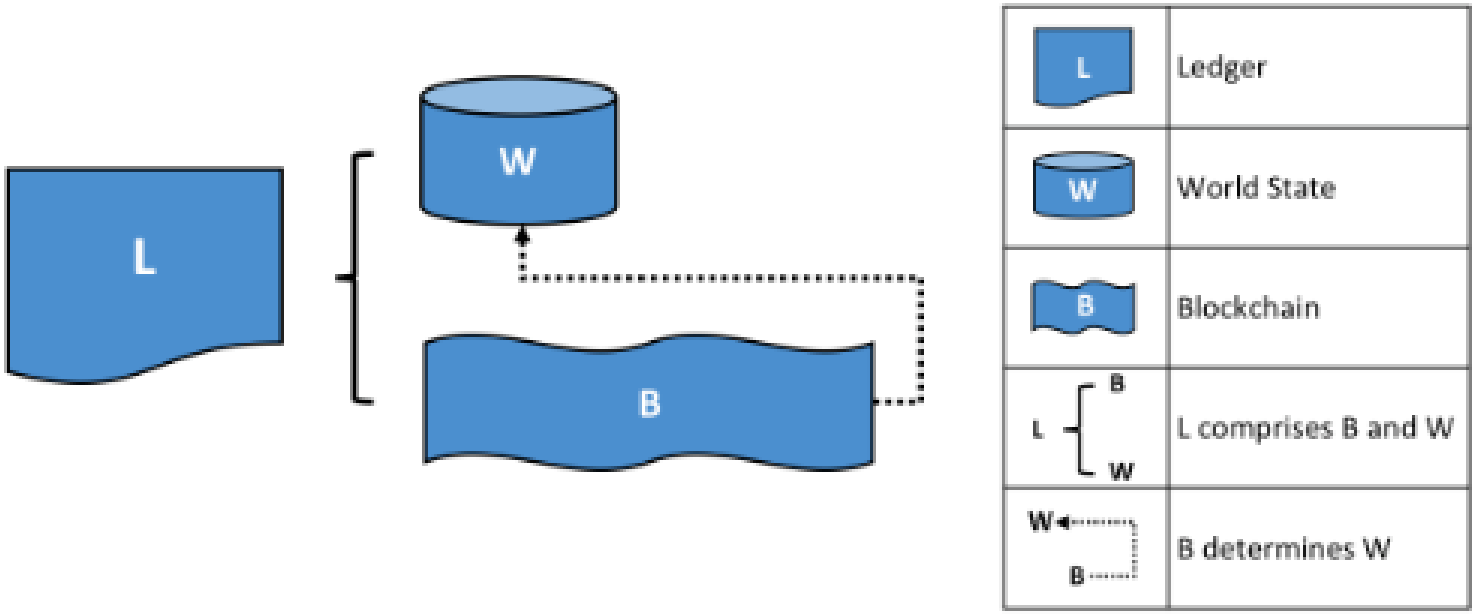
# Key Concepts

Introduction
Hyperledger Fabric Functionalities
Hyperledger Fabric Model
Blockchain network
Identity
Membership
Peers
Smart Contracts and Chaincode
Ledger
The Ordering Service
Private data
Channel capabilities
Use Cases

# Ledger

分类帐是Hyperledger Fabric的一个关键概念;它存储关于业务对象的重要事实信息;对象属性的当前值和产生这些当前值的交易历史记录

A **ledger** is a key concept in Hyperledger Fabric; it stores important factual information about business objects; both the **current value** of the attributes of the objects, and the **history of transactions** that resulted in these current values.



| | |
|---|---|
| L | Ledger |
| W | World State |
| B | Blockchain |
| L { B W | L comprises B and W |
| W ← B | B determines W |

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html

# Key Concepts

Introduction

Hyperledger Fabric Functionalities

Hyperledger Fabric Model

Blockchain network

Identity

Membership

Peers

Smart Contracts and Chaincode

Ledger

The Ordering Service

Private data

Channel capabilities

Use Cases

# The Ordering Service

许多分布式的区块链，如以太坊和比特币，是未许可的，这意味着任何节点都可以参与到共识过程中，其中的交易被排序和捆绑成块。

· 由于这一事实，这些系统依赖于概率一致性算法，这些算法最终保证分类帐在很大程度上的一致性，

· 但它们仍然容易受到不同分类账(也被称为账本分支)的影响，在分类账分支中，网络中的不同参与者对公认的交易顺序有不同的看法。

- Many distributed blockchains, such as Ethereum and Bitcoin, are not permissioned, which means that any node can participate in the consensus process, wherein transactions are ordered and bundled into blocks.
  - Because of this fact, these systems rely on **probabilistic** consensus algorithms which eventually guarantee ledger consistency to a high degree of probability,
  - but which are still vulnerable to divergent ledgers (also known as a ledger “fork”), where different participants in the network have a different view of the accepted order of transactions.

Hyperledger Fabric的工作原理不同。它提供了一种称为排序者的节点(也称为排序节点)，它执行交易排序，与其他节点一起形成了一个排序服务。

· 因为Fabric的设计依赖于确定性的一致算法，由排序服务生成的任何块都被确认为最终的和正确的。

· 账本不能像其他许多分布式区块链那样分叉

- Hyperledger Fabric works differently. It features a kind of a node called an **orderer** (it’s also known as an “ordering node”) that does this transaction ordering, which along with other nodes forms an **ordering service**.
  - Because Fabric’s design relies on **deterministic** consensus algorithms, any block a peer validates as generated by the ordering service is guaranteed to be final and correct.
  - Ledgers cannot fork the way they do in many other distributed blockchains.

除了促进终结性之外，将链码执行的认可(发生在对等节点上)与排序分离，可以使Fabric在性能和可伸缩性方面获得优势，消除在相同节点执行和排序时可能出现的瓶颈。

- In addition to promoting finality, separating the endorsement of chaincode execution (which happens at the peers) from ordering gives Fabric advantages in performance and scalability, eliminating bottlenecks which can occur when execution and ordering are performed by the same nodes.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html

# Ordering service implementations

虽然目前可用的每个排序服务都以相同的方式处理交易和配置更新，但仍有几种不同的实现可以在排序服务节点之间就严格的交易顺序达成共识

While every ordering service currently available handles transactions and configuration updates the same way, there are nevertheless several different implementations for achieving consensus on the strict ordering of transactions between ordering service nodes.

**Solo** 排序服务的单独实现的名称很恰当:它只提供一个排序节点。因此，它不是，也永远不会是容错的。由于这个原因，Solo实现不能用于生产，但是对于测试应用程序和智能契合，或者创建概念证明，它们是一个很好的选择。但是，如果您希望将这个PoC网络扩展到生产环境中，您可能希望从单个节点Raft集群开始，因为它可能会被重新配置以添加额外的节点

The Solo implementation of the ordering service is aptly named: it features only a single ordering node. As a result, it is not, and never will be, fault tolerant. For that reason, Solo implementations cannot be considered for production, but they are a good choice for testing applications and smart contracts, or for creating proofs of concept. However, if you ever want to extend this PoC network into production, you might want to start with a single node Raft cluster, as it may be reconfigured to add additional nodes.

**Raft**新的作为v1.4.1，Raft是一个崩溃容错(CFT)排序服务，基于Raft协议在etcd的实现。Raft遵循一个领导者和追随者模型，其中一个领导者节点被选出(每个通道)，其决策被追随者复制。与基于kafka的排序服务相比，Raft排序服务应该更容易建立和管理，而且它们的设计允许不同的组织为分布式排序服务贡献节点

New as of v1.4.1, Raft is a crash fault tolerant (CFT) ordering service based on an implementation of Raft protocol in etcd. Raft follows a “leader and follower” model, where a leader node is elected (per channel) and its decisions are replicated by the followers. Raft ordering services should be easier to set up and manage than Kafka-based ordering services, and their design allows different organizations to contribute nodes to a distributed ordering service.

**Kafka**类似于基于raft的排序，Apache Kafka是一个使用leader和follower节点配置的CFT实现。Kafka利用ZooKeeper团队进行管理。基于Kafka的排序服务从Fabric v1.0开始就可用了，但是许多用户可能会发现管理Kafka集群的额外管理开销令人生畏或不受欢迎

Similar to Raft-based ordering, Apache Kafka is a CFT implementation that uses a “leader and follower” node configuration. Kafka utilizes a ZooKeeper ensemble for management purposes. The Kafka based ordering service has been available since Fabric v1.0, but many users may find the additional administrative overhead of managing a Kafka cluster intimidating or undesirable.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html

# Key Concepts

Introduction
Hyperledger Fabric Functionalities
Hyperledger Fabric Model
Blockchain network
Identity
Membership
Peers
Smart Contracts and Chaincode
Ledger
The Ordering Service
Private data
Channel capabilities
Use Cases

# Private data

如果某个通道上的一组组织需要对该通道上的其他组织保持数据的私有，那么他们可以选择创建一个只包含需要访问数据的组织的新通道。
· 然而，在每种情况下创建单独的通道会产生额外的管理开销(维护链码版本、策略、MSPs等)，并且不允许在保持部分数据私有的情况下希望所有通道参与者看到交易。
· 这就是为什么从v1.2开始，Fabric提供了创建私有数据集合的能力，这允许在通道上定义的组织子集能够认可、提交或查询私有数据，而不必创建单独的通道

In cases where a group of organizations on a channel need to keep data private from other organizations on that channel, they have the option to create a **new channel** comprising just the organizations who need access to the data.

- However, creating separate channels in each of these cases creates additional administrative overhead (maintaining chaincode versions, policies, MSPs, etc), and doesn't allow for use cases in which you want all channel participants to see a transaction while keeping a portion of the data private.
- That's why, starting in v1.2, Fabric offers the ability to create **private data collections**, which allow a defined subset of organizations on a channel the ability to endorse, commit, or query private data without having to create a separate channel.

集合是两个元素的结合:
· 真实的私人数据，通过八卦协议点对点发送给授权的组织。
· 该数据存储在被授权组织的对等点上的私有状态数据库中(有时称为侧数据库或SideDB)，可以通过这些被授权的对等点上的链码访问该数据库。这里不涉及排序服务，也不会看到私有数据。
· 请注意，由于闲话将私有数据点对点地分布在被授权的组织中，因此需要在通道上设置锚点对等点，并在每个对等点上配置CORE_PEER_GOSSIP_EXTERNALENDPOINT，以便引导跨组织通信。
· 该数据的散列，被认可、排序并写入通道上每个对等点的总账。
· 哈希用作交易的证据，用于状态验证，也可用于审计目的。

## A collection is the combination of two elements:

- **Actual private data**, sent peer-to-peer via gossip protocol to only the organization(s) authorized to see it.
  - This data is stored in a private state database on the peers of authorized organizations (sometimes called a "side" database, or "SideDB"), which can be accessed from chaincode on these authorized peers. The ordering service is not involved here and does not see the private data.
  - Note that because gossip distributes the private data peer-to-peer across authorized organizations, it is required to set up **anchor peers** on the channel, and configure CORE_PEER_GOSSIP_EXTERNALENDPOINT on each peer, in order to bootstrap cross-organization communication.
- **A hash of that data**, which is endorsed, ordered, and written to the ledgers of every peer on the channel.
  - The hash serves as evidence of the transaction and is used for state validation and can be used for audit purposes.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/private-data/private-data.html

# Key Concepts

- Introduction
- Hyperledger Fabric Functionalities
- Hyperledger Fabric Model
- Blockchain network
- Identity
- Membership
- Peers
- Smart Contracts and Chaincode
- Ledger
- The Ordering Service
- Private data
- Channel capabilities
- Use Cases

# Channel capabilities

Fabric是一个分布式系统，通常会涉及多个组织，不同版本的Fabric代码可能(通常)存在于网络中的不同节点以及网络中的通道上。
· Fabric允许这样做，不必让每个对等节点和排序节点都处于相同的版本级别。
· 事实上，支持不同的版本级别正是支持Fabric节点滚动升级的原因。重要的是，网络和通道以同样的方式处理事情，为通道配置更新和链码调用等事情创建确定性的结果。
· 如果没有确定性结果，通道上的一个对等点可能会使交易无效，而另一个对等点可能会对其进行验证。

Fabric is a distributed system that will usually involve multiple organizations, it is possible (and typical) that **different versions** of Fabric code will exist on different nodes within the network as well as on the channels in that network.

  - Fabric allows this — it is not necessary for every peer and ordering node to be at the same version level.
  - In fact, supporting different version levels is what enables rolling upgrades of Fabric nodes. What **is** important is that networks and channels process things in the same way, creating deterministic results for things like channel configuration updates and chaincode invocations.
  - Without deterministic results, one peer on a channel might invalidate a transaction while another peer may validate it.

Fabric定义了所谓功能的级别。这些功能在每个通道的配置中定义，通过定义行为产生一致结果的级别，确保了确定性。
· 这些功能的版本与节点二进制版本密切相关。
· 功能使运行在不同版本级别的节点能够以一种兼容和一致的方式在特定块高度的通道配置中运行。
· 功能存在于配置树的许多部分中，这些部分沿着特定任务的管理线定义

- Fabric defines levels of what are called “capabilities”. These capabilities, which are defined in the configuration of each channel, ensure determinism by defining a level at which behaviors produce **consistent results**.
  - These capabilities have versions which are closely related to node binary versions.
  - Capabilities enable nodes running at different version levels to behave in a compatible and consistent way given the channel configuration at a specific block height.
  - Capabilities exist in many parts of the configuration tree, defined along the lines of administration for particular tasks.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/capabilities_concept.html

# Key Concepts

Introduction
Hyperledger Fabric Functionalities
Hyperledger Fabric Model
Blockchain network
Identity
Membership
Peers
Smart Contracts and Chaincode
Ledger
The Ordering Service
Private data
Channel capabilities
Use Cases

# Use Cases

| Use case Categories | | |
|---|---|---|
| Aerospace | Financial Services /Trade Finance | Land and Property Management |
| Cross Border Payments | Government /Public Sector | Letters Of Credit |
| Cross Border Payments | Green Assets Management | Real Estate Transactions |
| Digital Assets/ Identity Management | Healthcare | Smart Contracts |
| Education | Internet Of Things | Supply Chain Management |
| Energy | Information Technology | Ticketing /Entertainment |
| Food /Food Delivery | KYC | Voting |

**See details in:** https://wiki.hyperledger.org/display/LMDWG/Use+Cases

- Key Concepts
- Developing an Application

# Getting Started

·在我们开始之前，如果您还没有这样做，您可能希望检查一下，在您将要开发区块链应用程序和/或操作Hyperledger Fabric的平台上，您已经安装了所有的先决条件。
·一旦你安装好先决条件，你就可以下载并安装HyperLedger Fabric了。
·在为Fabric二进制文件开发真正的安装程序时，我们提供了一个脚本，用于将示例文件、二进制文件和Docker映像安装到您的系统中。该脚本还将把Docker镜像下载到本地注册表。

- Before we begin, if you haven’t already done so, you may wish to check that you have all the Prerequisites installed on the platform(s) on which you’ll be developing blockchain applications and/or operating Hyperledger Fabric.

- Once you have the prerequisites installed, you are ready to download and install HyperLedger Fabric.

- While we work on developing real installers for the Fabric binaries, we provide a script that will Install Samples, Binaries and Docker Images to your system. The script also will download the Docker images to your local registry.

# Contents

Prerequisites
Installations
Developing Applications
The scenario
Tutorials
Operations Guides
Commands Reference
Architecture Reference
Frequently Asked Questions

# Prerequisites (1/3)

## Install cURL

如果还没有安装cURL工具，或者在文档中运行cURL命令时出现错误，请下载该工具的最新版本。

Download the latest version of the cURL tool if it is not already installed or if you get errors running the curl commands from the documentation.

https://curl.haxx.se/download.html

## Docker and Docker Compose

您将需要在Hyperledger Fabric上操作或开发(或用于)的平台上安装以下产品:
· MacOSX， *nix或Windows 10: Docker需要Docker 17.06.2-ce或更高版本。
· 旧版本的Windows: Docker Toolbox–同样，需要Docker版本Docker 17.06.2-ce或更高。
您可以在终端提示符中使用以下命令检查已安装的Docker的版本

You will need the following installed on the platform on which you will be operating, or developing on (or for), Hyperledger Fabric:

- MacOSX, *nix, or Windows 10: Docker Docker version 17.06.2-ce or greater is required.
- Older versions of Windows: Docker Toolbox - again, Docker version Docker 17.06.2-ce or greater is required.

You can check the version of Docker you have installed with the following command from a terminal prompt:

```
docker --version
```

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html

# Prerequisites (2/3)

## Go Programming Language

Hyperledger Fabric的许多组件都使用了Go编程语言

Hyperledger Fabric uses the Go Programming Language for many of its components.

Go version 1.12.x is required.

考虑到我们将在Go中编写链式代码程序，有两个环境变量需要正确设置;您可以将这些设置放置在适当的启动文件中，从而使其永久。如果您正在使用Linux下的bash shell，则可以使用bashrc文件

Given that we will be writing chaincode programs in Go, there are two environment variables you will need to set properly; you can make these settings permanent by placing them in the appropriate startup file, such as your personal ~/.bashrc file if you are using the bash shell under Linux.

首先，必须将环境变量GOPATH设置为指向包含下载的Fabric代码基的Go工作区，使用类似的东西

- First, you must set the environment variable GOPATH to point at the Go workspace containing the downloaded Fabric code base, with something like:

```
export GOPATH=$HOME/go
```

其次，您应该(同样在适当的启动文件中)扩展命令搜索路径以包括Go bin目录，例如下面的示例用于Linux下的bash

- Second, you should (again, in the appropriate startup file) extend your command search path to include the Go bin directory, such as the following example for bash under Linux:

```
export PATH=$PATH:$GOPATH/bin
```

虽然这个目录在新的Go工作区安装中可能不存在，但是稍后Fabric构建系统会用构建系统的其他部分使用的少量Go可执行文件填充该目录。因此，即使您目前还没有这样的目录，也可以像上面那样扩展您的shell搜索路径

While this directory may not exist in a new Go workspace installation, it is populated later by the Fabric build system with a small number of Go executables used by other parts of the build system. So even if you currently have no such directory yet, extend your shell search path as above.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html

# Prerequisites (3/3)

如果您将利用Hyperledger Fabric SDK为Nodejs开发Hyperledger Fabric的应用程序，那么版本8就支持8.9.4或更高的版本。Nodejs版本10在10.15.3及更高版本得到支持。

## Node.js Runtime and NPM

If you will be developing applications for Hyperledger Fabric leveraging the Hyperledger Fabric SDK for Node.js, version 8 is supported from 8.9.4 and higher. Node.js version 10 is supported from 10.15.3 and higher.

Node.js download

默认情况下，Ubuntu 16.04附带Python 3.5.1作为python3二进制版本安装。为了成功完成npm安装操作，Fabric Nodejs SDK需要Python 2.7的迭代。使用以下命令检索2.7版本

## Python (The following applies to Ubuntu 16.04 users only.)

By default Ubuntu 16.04 comes with Python 3.5.1 installed as the `python3` binary. The Fabric Node.js SDK requires an iteration of Python 2.7 in order for `npm install` operations to complete successfully. Retrieve the 2.7 version with the following command:

```
sudo apt-get install python
```

Check your version(s):

```
python --version
```

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html

# Contents

Prerequisites
Installations
Developing Applications
The scenario
Tutorials
Operations Guides
Commands Reference
Architecture Reference
Frequently Asked Questions

# Install Samples, Binaries and Docker Images

确定机器上要放置fabric-samples存储库的位置，并在终端窗口中输入该目录。下面的命令将执行以下步骤:
1.如果需要，克隆hyperledger/fabric样本库
2.签出适当的版本标记
3.将Hyperledger Fabric特定平台的二进制文件和指定版本的配置文件安装到Fabric示例的/bin和/config目录中
4.下载指定版本的Hyperledger Fabric docker镜像
准备好之后，在将要安装Fabric示例和二进制文件的目录中，继续执行命令以下拉二进制文件和映像

Determine a location on your machine where you want to place the *fabric-samples* repository and enter that directory in a terminal window. The command that follows will perform the following steps:

1. If needed, clone the hyperledger/fabric-samples repository
2. Checkout the appropriate version tag
3. Install the Hyperledger Fabric platform-specific binaries and config files for the version specified into the /bin and /config directories of fabric-samples
4. Download the Hyperledger Fabric docker images for the version specified

Once you are ready, and in the directory into which you will install the Fabric Samples and binaries, go ahead and execute the command to pull down the binaries and images.

```
curl -sSL http://bit.ly/2ysbOFE | bash -s
```

The command above downloads and executes a bash script that will download and extract all of the platform-specific binaries you will need to set up your network and place them into the cloned repo you created above. It retrieves the following platform-specific binaries:

- `configtxgen`,
- `configtxlator`,
- `cryptogen`,
- `discover`,
- `idemixgen`
- `orderer`,
- `peer`, and
- `fabric-ca-client`

上面的命令下载并执行一个bash脚本，该脚本将下载并解压缩设置网络所需的所有特定于平台的二进制文件，并将它们放到上面创建的复制的repo中。它检索以下特定于平台的二进制文件:

并将它们放到当前工作目录的bin子目录中。您可能希望将其添加到PATH环境变量中，以便在不完全限定每个二进制文件的路径的情况下提取这些文件

最后，该脚本将把Hyperledger Fabric docker镜像从docker Hub下载到您的本地docker注册表中，并将它们标记为最新的。脚本在结束时列出安装的Docker映像。
看看每个镜像的名称;这些是最终将组成我们的Hyperledger Fabric的组成部分。您还会注意到，有相同镜像ID的两个实例—一个标记为amd64-1.x。x和一个标记为最新的。在1.2.0之前，下载的映像由uname -m确定，显示为x86_64-1.x.x

and places them in the `bin` sub-directory of the current working directory.
You may want to add that to your PATH environment variable so that these can be picked up without fully qualifying the path to each binary. e.g.:

```
export PATH=<path to download location>/bin:$PATH
```

Finally, the script will download the Hyperledger Fabric docker images from Docker Hub into your local Docker registry and tag them as 'latest'.
The script lists out the Docker images installed upon conclusion.
Look at the names for each image; these are the components that will ultimately comprise our Hyperledger Fabric network. You will also notice that you have two instances of the same image ID - one tagged as "amd64-1.x.x" and one tagged as "latest". Prior to 1.2.0, the image being downloaded was determined by `uname -m` and showed as "x86_64-1.x.x".

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/install.html

# Contents

Prerequisites

Installations

Developing Applications

The scenario

Tutorials

Operations Guides

Commands Reference

Architecture Reference

Frequently Asked Questions

# Developing Applications

本主题介绍了如何开发客户端应用程序和智能合约来解决使用Hyperledger Fabric的业务问题。在一个涉及多个组织的真实商业票据场景中，您将了解实现此目标所需的所有概念和任务。我们假设区块链网络已经可用

This topic covers how to develop a client application and smart contract to solve a business problem using Hyperledger Fabric. In a real world Commercial Paper scenario, involving multiple organizations, you'll learn about all the concepts and tasks required to accomplish this goal. We assume that the blockchain network is already available.

The topic is designed for multiple audiences:

- ◆ Solution and application architect
- ◆ Client application developer
- ◆ Smart contract developer
- ◆ Business professional

主题面向多个受众:

◆解决方案和应用架构师
◆客户端应用开发
◆智能合同开发商
◆业务专业

您可以选择按顺序阅读主题，也可以根据需要选择单独的部分。单独的主题部分是根据读者的相关性来标记的，所以无论你是在寻找商业信息还是技术信息，当一个主题适合你时，就会很清楚

You can chose to read the topic in order, or you can select individual sections as appropriate. Individual topic sections are marked according to reader relevance, so whether you're looking for business or technical information it'll be clear when a topic is for you.

本主题遵循一个典型的软件开发生命周期。它从业务需求开始，然后涵盖开发应用程序和智能合约以满足这些需求所需的所有主要技术活动。

The topic follows a typical software development lifecycle. It starts with business requirements, and then covers all the major technical activities required to develop an application and smart contract to meet these requirements.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/developing_applications.html

# Contents

Prerequisites

Installations

Developing Applications

The scenario

Tutorials

Operations Guides

Commands Reference

Architecture Reference

Frequently Asked Questions

# The scenario

在本主题中，我们将描述一个涉及六个使用PaperNet的组织的业务场景，这是一个建立在Hyperledger Fabric上的商业票据网络，用于发行、购买和赎回商业票据。我们将使用这个场景来概述参与者组织使用的商业票据应用程序和智能合同的开发需求

In this topic, we’re going to describe a business scenario involving six organizations who use PaperNet, a commercial paper network built on Hyperledger Fabric, to issue, buy and redeem commercial paper. We’re going to use the scenario to outline requirements for the development of commercial paper applications and smart contracts used by the participant organizations.

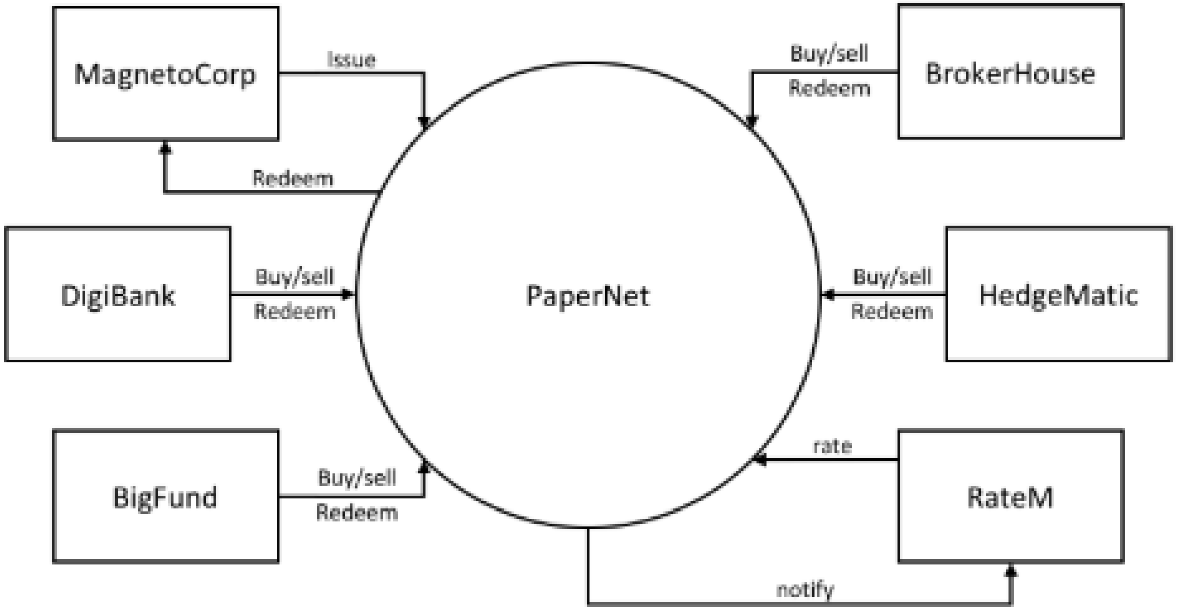
- **PaperNet network**
- **Introducing the actors**
- **Analysis**
- **Process and Data Design**
- **Smart Contract Processing**
- **Application**
- **Application design elements**

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/scenario.html

# PaperNet network

PaperNet是一个商业票据网络，允许适当授权的参与者发行、交易、赎回和评级商业票据

PaperNet is a commercial paper network that allows suitably authorized participants to issue, trade, redeem and rate commercial paper.



商业票据网。目前有六个机构使用PaperNet网络发行、购买、出售、赎回和评级商业票据。MagentoCorp发行和赎回商业票据。DigiBank、BigFund、BrokerHouse和HedgeMatic都相互进行商业票据交易。RateM为商业票据提供各种风险度量

*The PaperNet commercial paper network. Six organizations currently use PaperNet network to issue, buy, sell, redeem and rate commercial paper. MagentoCorp issues and redeems commercial paper. DigiBank, BigFund, BrokerHouse and HedgeMatic all trade commercial paper with each other. RateM provides various measures of risk for commercial paper.*

# Introducing the actors

- **MagnetoCorp** is a well-respected company that makes self-driving electric vehicles. In early April 2020, MagnetoCorp won a large order to manufacture 10,000 Model D cars for
- **Daintree**, a new entrant in the personal transport market. Although the order represents a significant win for MagnetoCorp, Daintree will not have to pay for the vehicles until they start to be delivered on November 1, six months after the deal was formally agreed between MagnetoCorp and Daintree.

To manufacture the vehicles, MagnetoCorp will need to hire 1000 workers for at least 6 months. This puts a short term strain on its finances – it will require an extra 5M USD each month to pay these new employees.

- **Commercial paper** is designed to help MagnetoCorp overcome its short term financing needs – to meet payroll every month based on the expectation that it will be cash rich when Daintree starts to pay for its new Model D cars.

At the end of May, MagnetoCorp needs 5M USD to meet payroll for the extra workers it hired on May 1. To do this, it issues a commercial paper with a face value of 5M USD with a maturity date 6 months in the future – when it expects to see cash flow from Daintree.

- **DigiBank** thinks that MagnetoCorp is creditworthy, and therefore doesn't require much of a premium above the central bank base rate of 2%, which would value 4.95M USD today at 5M USD in 6 months time. It therefore purchases the MagnetoCorp 6 month commercial paper for 4.94M USD – a slight discount compared to the 4.95M USD it is worth. DigiBank fully expects that it will be able to redeem 5M USD from MagnetoCorp in 6 months time, making it a profit of 10K USD for bearing the increased risk associated with this commercial paper. This extra 10K means it receives a 2.4% return on investment – significantly better than the risk free return of 2%.

At the end of June, when MagnetoCorp issues a new commercial paper for 5M USD to meet June's payroll, it is purchased by BigFund for 4.94M USD. That's because the commercial conditions are roughly the same in June as they are in May, resulting in BigFund valuing MagnetoCorp commercial paper at the same price that DigiBank did in May. Each subsequent month, MagnetoCorp can issue new commercial paper to meet its payroll obligations, and these may be purchased by DigiBank, or any other participant in the PaperNet commercial paper network –

- **BigFund**, **HedgeMatic** or **BrokerHouse**. These organizations may pay more or less for the commercial paper depending on two factors – the central bank base rate, and the risk associated with MagnetoCorp. This latter figure depends on a variety of factors such as the production of Model D cars, and the creditworthiness of MagnetoCorp as assessed by
- **RateM**, a ratings agency.

The organizations in PaperNet have different roles, MagnetoCorp issues paper, DigiBank, BigFund, HedgeMatic and BrokerHouse trade paper and RateM rates paper. Organizations of the same role, such as DigiBank, Bigfund, HedgeMatic and BrokerHouse are competitors. Organizations of different roles are not necessarily competitors, yet might still have opposing business interest, for example MagentoCorp will desire a high rating for its papers to sell them at a high price, while DigiBank would benefit from a low rating, such that it can buy them at a low price. As can be seen, even a seemingly simple network such as PaperNet can have complex trust relationships. A blockchain can help establish trust among organizations that are competitors or have opposing business interests that might lead to disputes. Fabric in particular has the means to capture even fine-grained trust relationships.

Let's pause the MagnetoCorp story for a moment, and develop the client applications and smart contracts that PaperNet uses to issue, buy, sell and redeem commercial paper as well as capture the trust relationships between the organizations. We'll come back to the role of the rating agency, RateM, a little later.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/scenario.html **or Introducing the actors.doc**

# Analysis (1/6)

Let’s analyze commercial paper in a little more detail. PaperNet participants such as MagnetoCorp and DigiBank use commercial paper transactions to achieve their business objectives – let’s examine the structure of a commercial paper and the transactions that affect it over time. We will also consider which organizations in PaperNet need to sign off on a transaction based on the trust relationships among the organizations in the network. Later we’ll focus on how money flows between buyers and sellers; for now, let’s focus on the first paper issued by MagnetoCorp.

## Commercial paper lifecycle

A paper 00001 is issued by MagnetoCorp on May 31. Spend a few moments looking at the first **state** of this paper, with its different properties and values:

Issuer = MagnetoCorp
Paper = 00001
Owner = MagnetoCorp
Issue date = 31 May 2020
Maturity = 30 November 2020
Face value = 5M USD
Current state = issued

This paper state is a result of the **issue** transaction and it brings MagnetoCorp’s first commercial paper into existence! Notice how this paper has a 5M USD face value for redemption later in the year. See how the Issuer and Owner are the same when paper 00001 is issued. Notice that this paper could be uniquely identified as MagnetoCorp00001 – a composition of the Issuer and Paper properties. Finally, see how the property Current state = issued quickly identifies the stage of MagnetoCorp paper 00001 in its lifecycle.

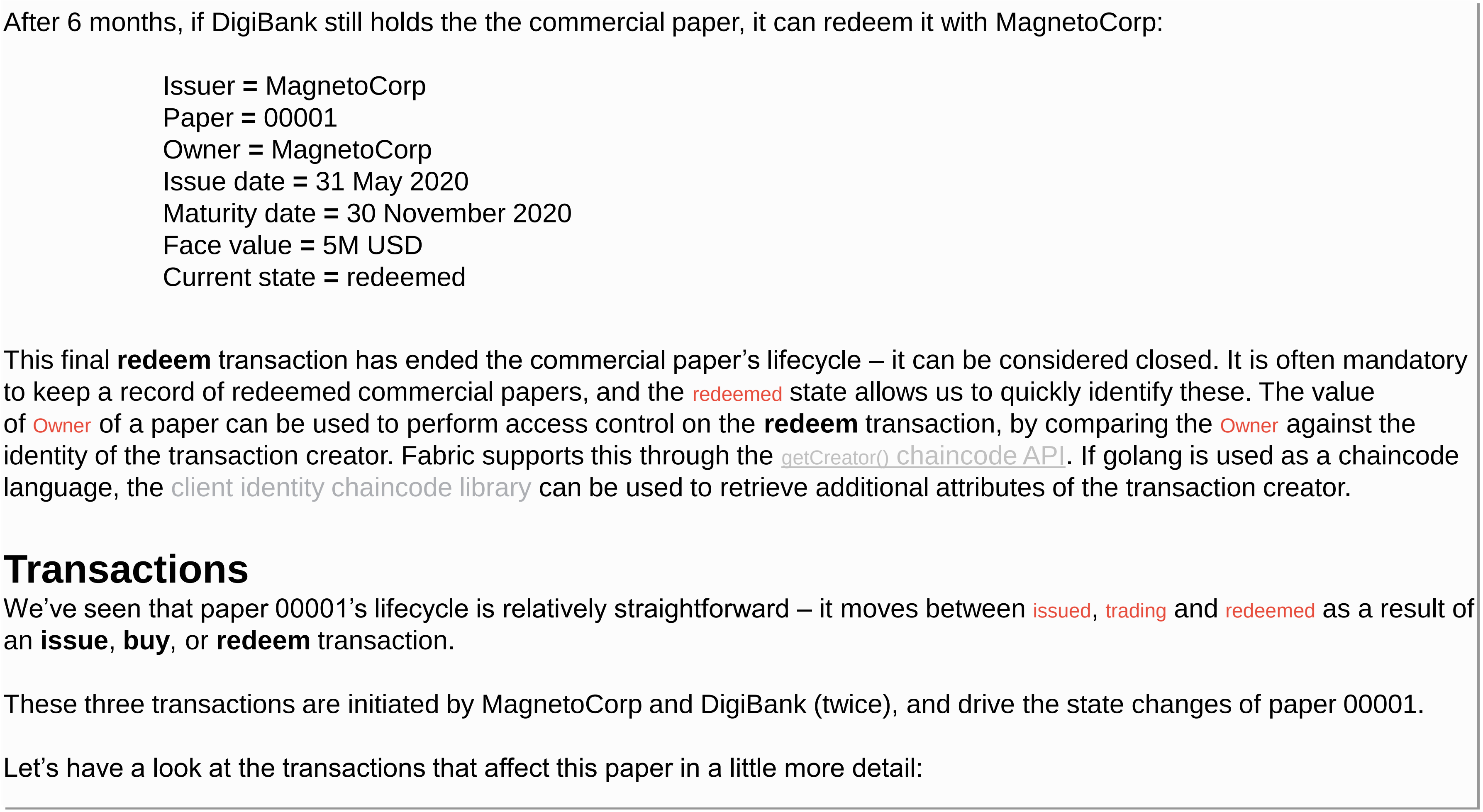
Shortly after issuance, the paper is bought by DigiBank. Spend a few moments looking at how the same commercial paper has changed as a result of this **buy** transaction:

Issuer = MagnetoCorp
Paper = 00001
Owner = MagnetoCorp
Issue date = 31 May 2020
Maturity date = 30 November 2020
Face value = 5M USD
Current state = redeemed

The most significant change is that of Owner – see how the paper initially owned by MagnetoCorp is now owned by DigiBank. We could imagine how the paper might be subsequently sold to BrokerHouse or HedgeMatic, and the corresponding change to Owner. Note how Current state allow us to easily identify that the paper is now trading.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Analysis (2/6)



After 6 months, if DigiBank still holds the the commercial paper, it can redeem it with MagnetoCorp:

Issuer = MagnetoCorp
Paper = 00001
Owner = MagnetoCorp
Issue date = 31 May 2020
Maturity date = 30 November 2020
Face value = 5M USD
Current state = redeemed

This final **redeem** transaction has ended the commercial paper's lifecycle – it can be considered closed. It is often mandatory to keep a record of redeemed commercial papers, and the redeemed state allows us to quickly identify these. The value of Owner of a paper can be used to perform access control on the **redeem** transaction, by comparing the Owner against the identity of the transaction creator. Fabric supports this through the getCreator() chaincode API. If golang is used as a chaincode language, the client identity chaincode library can be used to retrieve additional attributes of the transaction creator.

## Transactions

We've seen that paper 00001's lifecycle is relatively straightforward – it moves between issued, trading and redeemed as a result of an **issue**, **buy**, or **redeem** transaction.

These three transactions are initiated by MagnetoCorp and DigiBank (twice), and drive the state changes of paper 00001.

Let's have a look at the transactions that affect this paper in a little more detail:

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Analysis (3/6)

## Issue

Examine the first transaction initiated by MagnetoCorp:

Txn = issue
Issuer = MagnetoCorp
Paper = 00001
Issue time = 31 May 2020 09:00:00 EST
Maturity date = 30 November 2020
Face value = 5M USD

See how the **issue** transaction has a structure with properties and values. This transaction structure is different to, but closely matches, the structure of paper 00001. That's because they are different things – paper 00001 reflects a state of PaperNet that is a result of the **issue** transaction. It's the logic behind the **issue** transaction (which we cannot see) that takes these properties and creates this paper. Because the transaction **creates** the paper, it means there's a very close relationship between these structures.

The only organization that is involved in the **issue** transaction is MagnetoCorp. Naturally, MagnetoCorp needs to sign off on the transaction. In general, the issuer of a paper is required to sign off on a transaction that issues a new paper.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Analysis (4/6)

## Buy

Next, examine the **buy** transaction which transfers ownership of paper 00001 from MagnetoCorp to DigiBank:

Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = MagnetoCorp
New owner = DigiBank
Purchase time = 31 May 2020 10:00:00 EST
Price = 4.94M USD

See how the **buy** transaction has fewer properties that end up in this paper. That's because this transaction only **modifies** this paper. It's only New owner = DigiBank that changes as a result of this transaction; everything else is the same. That's OK – the most important thing about the **buy** transaction is the change of ownership, and indeed in this transaction, there's an acknowledgement of the current owner of the paper, MagnetoCorp.

You might ask why the Purchase time and Price properties are not captured in paper 00001? This comes back to the difference between the transaction and the paper. The 4.94 M USD price tag is actually a property of the transaction, rather than a property of this paper. Spend a little time thinking about this difference; it is not as obvious as it seems.

We're going to see later that the ledger will record both pieces of information – the history of all transactions that affect this paper, as well its latest state. Being clear on this separation of information is really important. parties that are part of the deal.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Analysis (5/6)

It’s also worth remembering that paper 00001 may be bought and sold many times. Although we’re skipping ahead a little in our scenario, let’s examine what transactions we **might** see if paper 00001 changes ownership.
If we have a purchase by BigFund:

```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = DigiBank
New owner = BigFund
Purchase time = 2 June 2020 12:20:00 EST
Price = 4.93M USD
```

Followed by a subsequent purchase by HedgeMatic:

```
Txn = buy
Issuer = MagnetoCorp
Paper = 00001
Current owner = BigFund
New owner = HedgeMatic
Purchase time = 3 June 2020 15:59:00 EST
Price = 4.90M USD
```

See how the paper owners changes, and how in out example, the price changes. Can you think of a reason why the price of MagnetoCorp commercial paper might be falling?
Intuitively, a **buy** transaction demands that both the selling as well as the buying organization need to sign off on such a transaction such that there is proof of the mutual agreement among the two parties that are part of the deal.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Analysis (6/6)

## Redeem

The **redeem** transaction for paper 00001 represents the end of its lifecycle. In our relatively simple example, HedgeMatic initiates the transaction which transfers the commercial paper back to MagnetoCorp:

Txn = redeem
Issuer = MagnetoCorp
Paper = 00001
Current owner = HedgeMatic
Redeem time = 30 Nov 2020 12:00:00 EST

Again, notice how the **redeem** transaction has very few properties; all of the changes to paper 00001 can be calculated data by the redeem transaction logic: the Issuer will become the new owner, and the Current state will change to redeemed. The Current owner property is specified in our example, so that it can be checked against the current holder of the paper.
From a trust perspective, the same reasoning of the **buy** transaction also applies to the **redeem** instruction: both organizations involved in the transaction are required to sign off on it.

## The Ledger

In this topic, we've seen how transactions and the resultant paper states are the two most important concepts in PaperNet. Indeed, we'll see these two fundamental elements in any Hyperledger Fabric distributed ledger – a world state, that contains the current value of all objects, and a blockchain that records the history of all transactions that resulted in the current world state.
The required sign-offs on transactions are enforced through rules, which are evaluated before appending a transaction to the ledger. Only if the required signatures are present, Fabric will accept a transaction as valid.

You're now in a great place translate these ideas into a smart contract. Don't worry if your programming is a little rusty, we'll provide tips and pointers to understand the program code. Mastering the commercial paper smart contract is the first big step towards designing your own application. Or, if you're a business analyst who's comfortable with a little programming, don't be afraid to keep dig a little deeper!

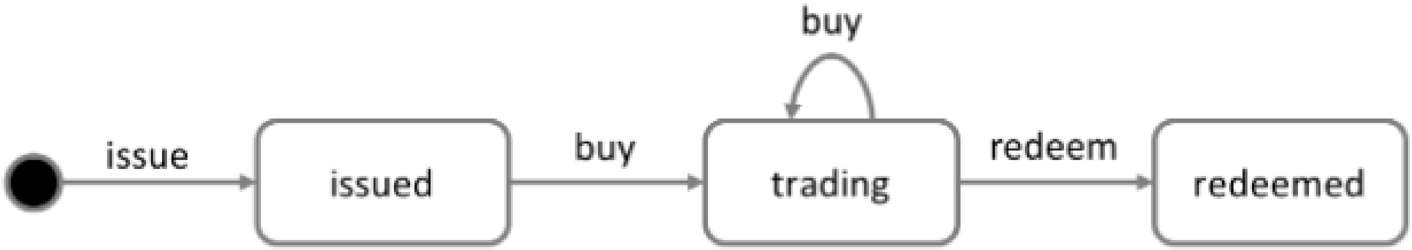
**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/analysis.html **or Analysis.doc**

# Process and Data Design (1/7)

## Lifecycle

As we've seen, there are two important concepts that concern us when dealing with commercial paper; states and transactions. Indeed, this is true for all blockchain use cases; there are conceptual objects of value, modeled as states, whose lifecycle transitions are described by transactions. An effective analysis of states and transactions is an essential starting point for a successful implementation.

We can represent the life cycle of a commercial paper using a state transition diagram:



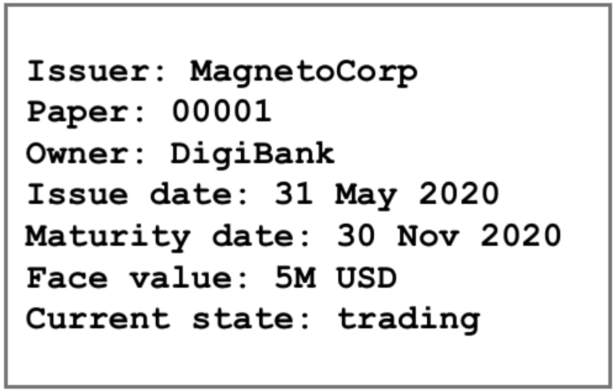
*The state transition diagram for commercial paper. Commercial papers transition between **issued**, **trading** and **redeemed** states by means of the **issue**, **buy** and **redeem** transactions.*

See how the state diagram describes how commercial papers change over time, and how specific transactions govern the life cycle transitions. In Hyperledger Fabric, smart contracts implement transaction logic that transition commercial papers between their different states. Commercial paper states are actually held in the ledger world state; so let's take a closer look at them.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.doc

# Process and Data Design (2/7)

Recall the structure of a commercial paper:



```
Issuer: MagnetoCorp
Paper: 00001
Owner: DigiBank
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: trading
```

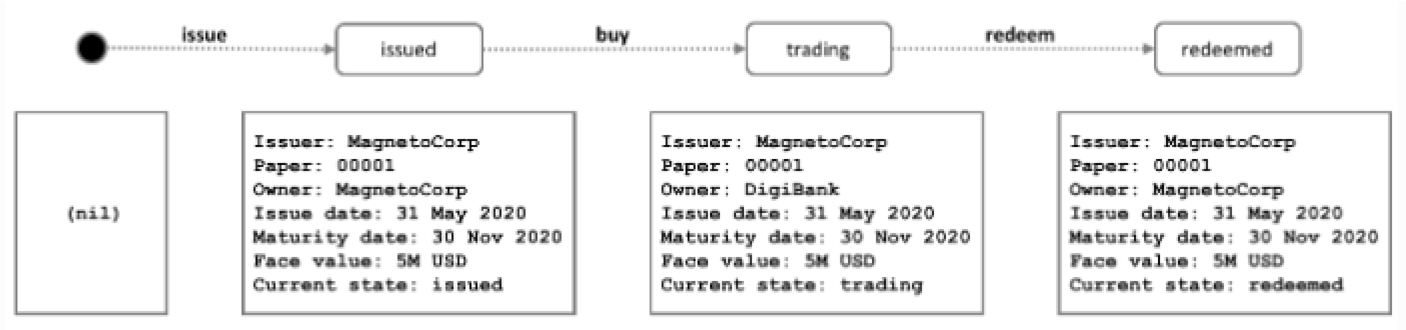
*A commercial paper can be represented as a set of properties, each with a value. Typically, some combination of these properties will provide a unique key for each paper.*

See how a commercial paper Paper property has value 00001, and the Face value property has value 5M USD. Most importantly, the Current state property indicates whether the commercial paper is issued, trading or redeemed. In combination, the full set of properties make up the **state** of a commercial paper. Moreover, the entire collection of these individual commercial paper states constitutes the ledger world state.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.**doc**

# Process and Data Design (3/7)

All ledger state share this form; each has a set of properties, each with a different value. This *multi-property* aspect of states is a powerful feature – it allows us to think of a Fabric state as a vector rather than a simple scalar. We then represent facts about whole objects as individual states, which subsequently undergo transitions controlled by transaction logic. A Fabric state is implemented as a key/value pair, in which the value encodes the object properties in a format that captures the object's multiple properties, typically JSON. The ledger database can support advanced query operations against these properties, which is very helpful for sophisticated object retrieval.
See how MagnetoCorp's paper 00001 is represented as a state vector that transitions according to different transaction stimuli:



(nil)

Issuer: MagnetoCorp
Paper: 00001
Owner: MagnetoCorp
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: issued

Issuer: MagnetoCorp
Paper: 00001
Owner: DigiBank
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: trading

Issuer: MagnetoCorp
Paper: 00001
Owner: MagnetoCorp
Issue date: 31 May 2020
Maturity date: 30 Nov 2020
Face value: 5M USD
Current state: redeemed

*A commercial paper state is brought into existence and transitions as a result of different transactions. Hyperledger Fabric states have multiple properties, making them vectors rather than scalars.*

Notice how each individual paper starts with the empty state, which is technically a nil state for the paper, as it doesn't exist! See how paper 00001 is brought into existence by the **issue** transaction, and how it is subsequently updated as a result of the **buy** and **redeem** transactions.

Notice how each state is self-describing; each property has a name and a value. Although all our commercial papers currently have the same properties, this need not be the case for all time, as Hyperledger Fabric supports different states having different properties. This allows the same ledger world state to contain different forms of the same asset as well as different types of asset. It also makes it possible to update a state's structure; imagine a new regulation that requires an additional data field. Flexible state properties support the fundamental requirement of data evolution over time.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.**doc**

# Process and Data Design (4/7)

## State keys

In most practical applications, a state will have a combination of properties that uniquely identify it in a given context – it's **key**. The key for a PaperNet commercial paper is formed by a concatenation of the Issuer and paper properties; so for MagnetoCorp's first paper, it's MagnetoCorp00001.

➢A state key allows us to uniquely identify a paper; it is created as a result of the **issue** transaction and subsequently updated by **buy** and **redeem**. Hyperledger Fabric requires each state in a ledger to have a unique key.

➢When a unique key is not available from the available set of properties, an application-determined unique key is specified as an input to the transaction that creates the state. This unique key is usually with some form of UUID, which although less readable, is a standard practice. What's important is that every individual state object in a ledger must have a unique key. *Note: You should avoid using U+0000 (nil byte) in keys.*

## Multiple states

As we've seen, commercial papers in PaperNet are stored as state vectors in a ledger. It's a reasonable requirement to be able to query different commercial papers from the ledger; for example: find all the papers issued by MagnetoCorp, or: find all the papers issued by MagnetoCorp in the redeemed state.

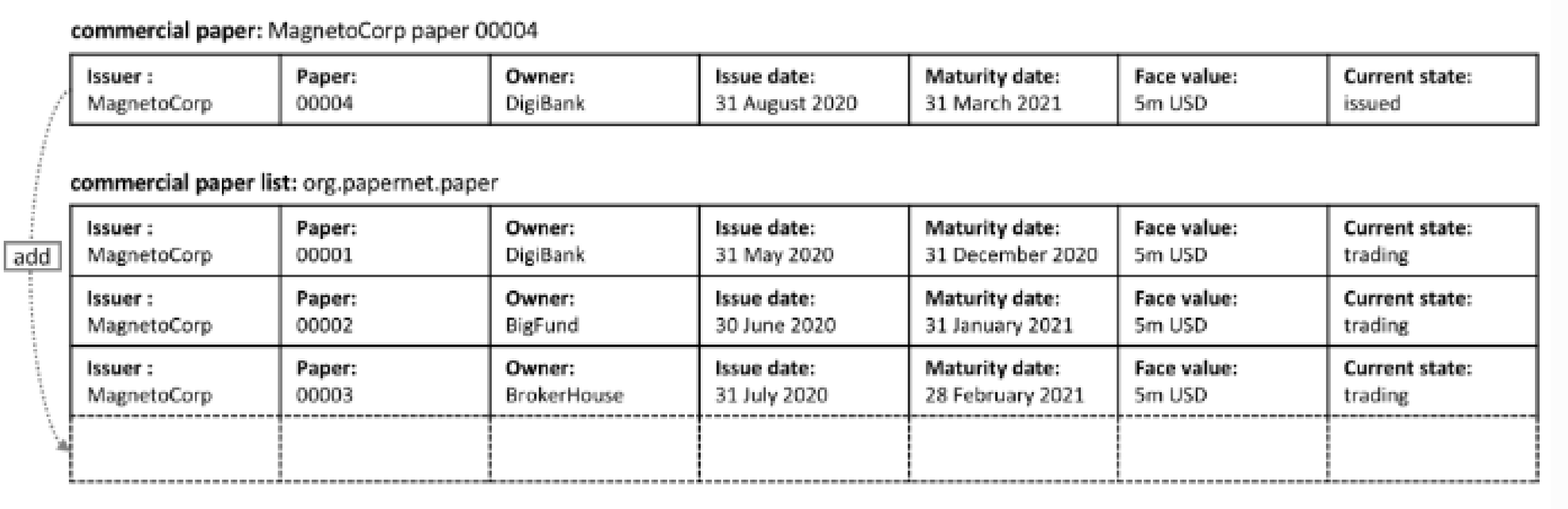
To make these kinds of search tasks possible, it's helpful to group all related papers together in a logical list. The PaperNet design incorporates the idea of a commercial paper list – a logical container which is updated whenever commercial papers are issued or otherwise changed.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.doc

# Process and Data Design (5/7)

**Logical representation**

It’s helpful to think of all PaperNet commercial papers being in a single list of commercial papers:



**commercial paper:** MagnetoCorp paper 00004

| Issuer : | Paper: | Owner: | Issue date: | Maturity date: | Face value: | Current state: |
|---|---|---|---|---|---|---|
| MagnetoCorp | 00004 | DigiBank | 31 August 2020 | 31 March 2021 | 5m USD | issued |

**commercial paper list:** org.papernet.paper

| Issuer : | Paper: | Owner: | Issue date: | Maturity date: | Face value: | Current state: |
|---|---|---|---|---|---|---|
| MagnetoCorp | 00001 | DigiBank | 31 May 2020 | 31 December 2020 | 5m USD | trading |
| MagnetoCorp | 00002 | BigFund | 30 June 2020 | 31 January 2021 | 5m USD | trading |
| MagnetoCorp | 00003 | BrokerHouse | 31 July 2020 | 28 February 2021 | 5m USD | trading |
| | | | | | | |

*MagnetoCorp’s newly created commercial paper 00004 is added to the list of existing commercial papers.*

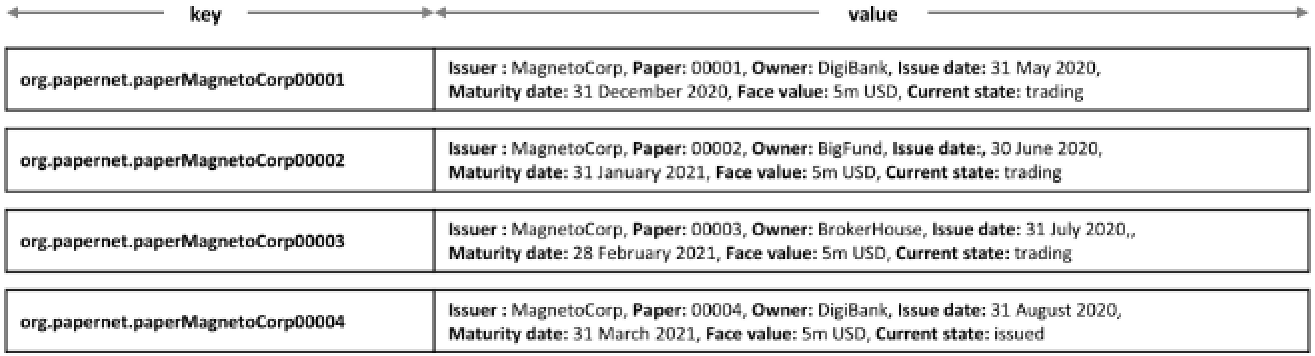
New papers can be added to the list as a result of an **issue** transaction, and papers already in the list can be updated with **buy** or **redeem** transactions. See how the list has a descriptive name: org.papernet.papers; it’s a really good idea to use this kind of DNS name because well-chosen names will make your blockchain designs intuitive to other people. This idea applies equally well to smart contract names.

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.**doc**

# Process and Data Design (6/7)

### Physical representation

While it’s correct to think of a single list of papers in PaperNet – org.papernet.papers – lists are best implemented as a set of individual Fabric states, whose composite key associates the state with its list. In this way, each state’s composite key is both unique and supports effective list query.



| key | value |
|---|---|
| **org.papernet.paperMagnetoCorp00001** | **Issuer :** MagnetoCorp, **Paper:** 00001, **Owner:** DigiBank, **Issue date:** 31 May 2020, **Maturity date:** 31 December 2020, **Face value:** 5m USD, **Current state:** trading |
| **org.papernet.paperMagnetoCorp00002** | **Issuer :** MagnetoCorp, **Paper:** 00002, **Owner:** BigFund, **Issue date:**, 30 June 2020, **Maturity date:** 31 January 2021, **Face value:** 5m USD, **Current state:** trading |
| **org.papernet.paperMagnetoCorp00003** | **Issuer :** MagnetoCorp, **Paper:** 00003, **Owner:** BrokerHouse, **Issue date:** 31 July 2020,, **Maturity date:** 28 February 2021, **Face value:** 5m USD, **Current state:** trading |
| **org.papernet.paperMagnetoCorp00004** | **Issuer :** MagnetoCorp, **Paper:** 00004, **Owner:** DigiBank, **Issue date:** 31 August 2020, **Maturity date:** 31 March 2021, **Face value:** 5m USD, **Current state:** issued |

*Representing a list of PaperNet commercial papers as a set of distinct Hyperledger Fabric states*

Notice how each paper in the list is represented by a vector state, with a unique **composite** key formed by the concatenation of org.papernet.paper, Issuer and Paper properties. This structure is helpful for two reasons:

➢It allows us to examine any state vector in the ledger to determine which list it’s in, without reference to a separate list. It’s analogous to looking at set of sports fans, and identifying which team they support by the colour of the shirt they are wearing. The sports fans self-declare their allegiance; we don’t need a list of fans.

➢Hyperlegder Fabric internally uses a concurrency control mechanism to update a ledger, such that keeping papers in separate state vectors vastly reduces the opportunity for shared-state collisions. Such collisions require transaction re-submission, complicate application design, and decrease performance.

This second point is actually a key take-away for Hyperledger Fabric; the physical design of state vectors is **very important** to optimum performance and behavior. Keep your states separate!

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.doc

# Process and Data Design (7/7)

**Trust relationships**

We have discussed how the different roles in a network, such as issuer, trader or rating agencies as well as different business interests determine who needs to sign off on a transaction.

In Fabric, these rules are captured by so-called **endorsement policies**. The rules can be set on a chaincode granularity, as well as for individual state keys.

This means that in PaperNet, we can set one rule for the whole namespace that determines which organizations can issue new papers.

Later, rules can be set and updated for individual papers to capture the trust relationships of buy and redeem transactions.

In the next topic, we will show you how to combine these design concepts to implement the PaperNet commercial paper smart contract, and then an application in exploits it!

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/architecture.html or **Process and Data Design**.doc

# Smart Contract Processing

At the heart of a blockchain network is a smart contract.
In PaperNet, the code in the commercial paper smart contract defines the valid states for commercial paper, and the transaction logic that transition a paper from one state to another. In this topic, we're going to show you how to implement a real world smart contract that governs the process of issuing, buying and redeeming commercial paper. We're going to cover:

- What is a smart contract and why it's important
- How to define a smart contract
- How to define a transaction
- How to implement a transaction
- How to represent a business object in a smart contract
- How to store and retrieve an object in the ledger

If you'd like, you can download the sample and even run it locally. It is written in JavaScript and Java, but the logic is quite language independent, so you'll be easily able to see what's going on! (The sample will become available for Go as well.)

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/smartcontract.html or **Smart Contract Processing.doc**

# Application

An application can interact with a blockchain network by submitting transactions to a ledger or querying ledger content. This topic covers the mechanics of how an application does this; in our scenario, organizations access PaperNet using applications which invoke **issue**, **buy** and **redeem** transactions defined in a commercial paper smart contract. Even though MagnetoCorp's application to issue a commercial paper is basic, it covers all the major points of understanding.

In this topic, we're going to cover:

- The application flow to invoke a smart contract
- How an application uses a wallet and identity
- How an application connects using a gateway
- How to access a particular network
- How to construct a transaction request
- How to submit a transaction
- How to process a transaction response

To help your understanding, we'll make reference to the commercial paper sample application provided with Hyperledger Fabric. You can download it and run it locally.

- It is written in both JavaScript and Java, but the logic is quite language independent, so you'll be easily able to see what's going on! (The sample will become available for Go as well.)

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/application.html or **Application.doc**

# Application design elements

This section elaborates the key features for client application and smart contract development found in Hyperledger Fabric.

A solid understanding of the features will help you design and implement efficient and effective solutions.

- Contract names
- Chaincode namespace
- Transaction context
- Transaction handlers
- Endorsement policies
- Connection Profile
- Connection Options
- Wallet
- Gateway

**See details in:** https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/designelements.html **or Application design elements.doc**

# Contents

**Prerequisites**
**Installations**
**Developing Applications**
**The scenario**
**Tutorials**
**Operations Guides**
**Commands Reference**
**Architecture Reference**
**Frequently Asked Questions**

# Tutorials

我们提供教程，让你开始与Hyperledger Fabric。

·第一个面向Hyperledger Fabric应用程序开发人员，编写您的第一个应用程序。它带您完成使用Hyperledger Fabric节点SDK为Hyperledger Fabric编写第一个区块链应用程序的过程。

·第二个教程是面向Hyperledger Fabric运营商，建立你的第一个网络。本文带您完成了使用Hyperledger Fabric建立区块链网络的过程，并提供了一个基本的示例应用程序来测试它。

·还有一些教程可以更新你的通道，为通道添加一个组织，升级你的网络到一个更高版本的Hyperledger结构，升级你的网络组件。

·最后，我们提供了两个链码教程。一个面向开发者，链码面向开发者，另一个面向操作员，链码面向操作员。

## We offer tutorials to get you started with Hyperledger Fabric.

- The first is oriented to the Hyperledger Fabric application **developer**, Writing Your **First Application**. It takes you through the process of writing your first blockchain application for Hyperledger Fabric using the Hyperledger Fabric Node SDK.

- The second tutorial is oriented towards the Hyperledger Fabric network **operators**, Building Your **First Network**. This one walks you through the process of establishing a blockchain network using Hyperledger Fabric and provides a basic sample application to test it out.

- There are also tutorials for updating your channel, Adding an Org to a Channel, and upgrading your network to a later version of Hyperledger Fabric, Upgrading Your Network Components.

- Finally, we offer two chaincode tutorials. One oriented to developers, Chaincode for Developers, and the other oriented to operators, Chaincode for Operators.
  - ◆ **Writing Your First Application**
  - ◆ **Building Your First Network**
  - ◆ **Commercial paper tutorial**
  - ◆ **Updating a channel configuration**
  - ◆ **Adding an Org to a Channel**
  - ◆ **Upgrading Your Network Components**
  - ◆ **Using Private Data in Fabric**
  - ◆ **Chaincode Tutorials**
    - ◆ **Chaincode for Developers**
    - ◆ **Chaincode for Operators**
  - ◆ **Writing Your First Chaincode**
  - ◆ **Deploying a production network**
  - ◆ **Using CouchDB**
  - ◆ **Videos**

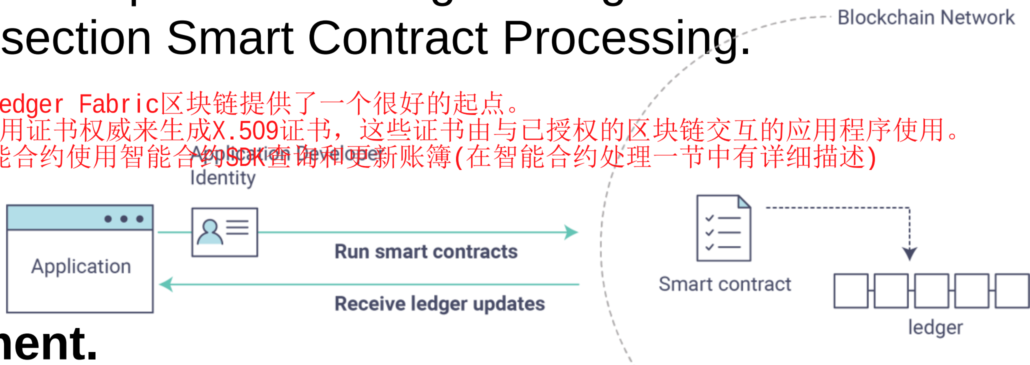
# Writing Your First Application

We'll be looking at a handful of sample programs to see how Fabric apps work.

- These applications and the smart contracts they use are collectively known as FabCar. They provide a great starting point to understand a Hyperledger Fabric blockchain.
- You'll learn how to write an application and smart contract to query and update a ledger, and how to use a Certificate Authority to generate the X.509 certificates used by applications which interact with a permissioned blockchain.

We will use the application SDK — described in detail in the Application topic – to invoke a smart contract which queries and updates the ledger using the smart contract SDK — described in detail in section Smart Contract Processing.

我们将通过一些示例程序来了解Fabric应用程序是如何工作的。
· 这些应用程序和它们使用的智能合同统称为FabCar。它们为理解Hyperledger Fabric区块链提供了一个很好的起点。
· 将学习如何编写应用程序和智能合约来查询和更新分类帐，以及如何使用证书权威来生成X.509证书，这些证书由与已授权的区块链交互的应用程序使用。
我们将使用应用SDK(在应用主题中有详细描述)调用一个智能合约，该智能合约使用智能合约SDK查询和更新账簿(在智能合约处理一节中有详细描述)



We'll go through three principle steps:

**1.Setting up a development environment.**
**2.Learning about a sample smart contract, FabCar.**
**3.Develop a sample application which uses FabCar.**

我们将经历三个主要步骤:
1.设置开发环境。
2.学习一个智能合同的例子，FabCar。
3.开发一个使用FabCar的示例应用程序

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html **or Writing Your First Application.DOC**

# Building Your First Network

构建第一个网络(BYFN)场景提供了一个示例Hyperledger Fabric网络，它由两个组织组成，每个组织维护两个对等节点。
默认情况下，它还将部署一个单独的排序服务，不过其他排序服务的实现也是可用的

The build your first network (BYFN) scenario provisions a sample Hyperledger Fabric network consisting of two organizations, each maintaining two peer nodes.

It also will deploy a “Solo” ordering service by default, though other ordering service implementations are available.

- If you are getting started with Hyperledger Fabric and would like to deploy a **basic network**, see Using the Fabric test network.
- If you are deploying Fabric **in production**, see the guide for Deploying a production network.

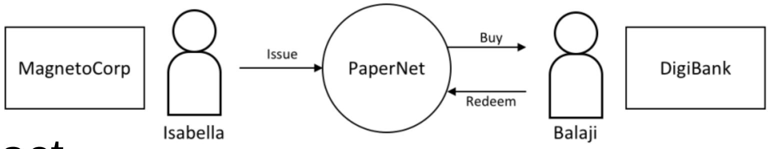
**See details in： https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html or Building Your First Network.DOC**

您将扮演不同组织中的开发人员、终端用户和管理员，执行以下步骤，旨在帮助您理解作为两个独立工作的不同组织，但根据Hyperledger Fabric网络中相互同意的规则进行协作的感觉。

- 设置机器并下载示例
- 创建一个网络
- 理解智能合同的结构
- 作为一个组织，MagnetoCorp，安装和实例化智能合同
- 理解MagnetoCorp应用程序的结构，包括它的依赖关系
- 配置和使用钱包和身份
- 运行磁石申请发行商业票据
- 了解第二个组织Digibank如何在其应用程序中使用智能合同
- 作为Digibank，它运行购买和赎回商业票据的应用程序

# Commercial paper tutorial

You'll act as an developer, end user, and administrator, each in different organizations, performing the following steps designed to help you understand what it's like to collaborate as two different organizations working independently, but according to mutually agreed rules in a Hyperledger Fabric network.



- ➢ Set up machine and download samples
- ➢ Create a network
- ➢ Understand the structure of a smart contract
- ➢ Work as an organization, MagnetoCorp, to install and instantiate smart contract
- ➢ Understand the structure of a MagnetoCorp application, including its dependencies
- ➢ Configure and use a wallet and identities
- ➢ Run a MagnetoCorp application to issue a commercial paper
- ➢ Understand how a second organization, Digibank, uses the smart contract in their applications
- ➢ As Digibank, run applications that buy and redeem commercial paper

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/tutorial/commercial_paper.html **or Commercial paper tutorial.DOC**

# Updating a channel configuration

In this topic, we’ll:

在这个主题:
· 显示应用程序通道的完整样例配置。
· 讨论许多可以编辑的通道参数。
· 展示更新通道配置的过程，包括将配置拉出、转换和作用域为人们可以读取的内容所必需的命令。
· 讨论可用于编辑通道配置的方法。
· 显示用于重新格式化配置的流程，并获取批准配置所需的签名。

- Show a full sample configuration of an application channel.
- Discuss many of the channel parameters that can be edited.
- Show the process for updating a channel configuration, including the commands necessary to pull, translate, and scope a configuration into something that humans can read.
- Discuss the methods that can be used to edit a channel configuration.
- Show the process used to reformat a configuration and get the signatures necessary for it to be approved.

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/config_update.html **or Updating a channel configuration.DOC**

# Adding an Org to a Channel

本教程作为构建第一个网络(BYFN)教程的扩展，并将演示如何向BYFN自动生成的应用程序通道(mychannel)添加一个新的组织Org3。
本文假设您非常了解BYFN，包括前面提到的实用程序的用法和功能

This tutorial serves as an extension to the Building Your First Network (BYFN) tutorial, and will demonstrate the addition of a new organization – Org3 – to the application channel (mychannel) autogenerated by BYFN.

It assumes a strong understanding of BYFN, including the usage and functionality of the aforementioned utilities.

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/channel_update_tutorial.html **or Adding an Org to a Channel.DOC**

# Upgrading Your Network Components

This tutorial will perform the following steps:

- Backup the ledger and MSPs.
- Upgrade the orderer binaries to Fabric v1.4.x.
- Upgrade the peer binaries to Fabric v1.4.x.
- Update channel capabilities to v1.4.2 and 1.4.3 (optional).

本教程将演示如何使用CLI命令单独执行这些步骤。包括脚本执行和手动执行的说明

This tutorial will demonstrate how to perform each of these steps individually with CLI commands. Instructions for both scripted execution and manual execution are included.

**See details in：** https://hyperledger-fabric.readthedocs.io/en/release-1.4/upgrading_your_network_tutorial.html **or Upgrading Your Network Components.DOC**

# Using Private Data in Fabric

The tutorial will take you through the following steps to practice defining, configuring and using private data with Fabric:

- Build a collection definition JSON file
- Read and Write private data using chaincode APIs
- Install and instantiate chaincode with a collection
- Store private data
- Query the private data as an authorized peer
- Query the private data as an unauthorized peer
- Purge Private Data
- Using indexes with private data
- Additional resources

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/private_data_tutorial.html **or Using Private Data in Fabric.DOC**

· 链码是一个程序，用Go、node.js或java编写,以实现指定接口
· 链码运行在一个安全的Docker容器中，与认证对等进程隔离。
· 链码通过应用程序提交的交易来初始化和管理分类账状态。
链码通常处理网络成员同意的业务逻辑，因此可以认为是一种智能合约。
· 由链码创建的状态仅作用域为该链码，不能被另一个链码直接访问。
· 然而，在同一网络中，给定适当的权限，一个链码可以调用另一个链码来访问其状态

# Chaincode Tutorials

- Chaincode is a program, written in Go, node.js, or Java that implements a prescribed interface.
- Chaincode runs in a secured Docker container isolated from the endorsing peer process.
- Chaincode initializes and manages ledger state through transactions submitted by applications.

A chaincode typically handles business logic agreed to by members of the network, so it may be considered as a “smart contract”.

- State created by a chaincode is scoped exclusively to that chaincode and can’t be accessed directly by another chaincode.
- However, within the same network, given the appropriate permission a chaincode may invoke another chaincode to access its state.

关于链码，我们提供两种不同的视角：
· 第一，从一个为开发者开发名为链码的区块链应用/解决方案的应用开发者的角度来看，
· 另一种是面向区块链网络运营商的链码，该网络运营商负责管理区块链网络，并利用Hyperledger Fabric API来安装、实例化和升级链码，但可能不会参与链码应用程序的开发

**We offer two different perspectives on chaincode：**

- One, from the perspective of an application developer developing a blockchain application/solution entitled Chaincode for Developers, and
- the other, Chaincode for Operators oriented to the blockchain network operator who is responsible for managing a blockchain network, and who would leverage the Hyperledger Fabric API to install, instantiate, and upgrade chaincode, but would likely not be involved in the development of a chaincode application.

**See details in：** https://hyperledger-fabric.readthedocs.io/en/release/chaincode.html **or Chaincode Tutorials.DOC**

# Chaincode for Developers

- Chaincode API
- Simple Asset Chaincode
  - Choosing a Location for the Code
  - Housekeeping
  - Initializing the Chaincode
  - Invoking the Chaincode
  - Implementing the Chaincode Application
  - Pulling it All Together
- Chaincode access control
- Managing external dependencies for chaincode written in Go

**See details in： https://hyperledger-fabric.readthedocs.io/en/latest/chaincode4ade.html or Chaincode for Developers.DOC**

# Chaincode for Operators

- Chaincode lifecycle
- Packaging
- Creating the package
- Package signing
- Installing chaincode
- Instantiate
- Upgrade
- Stop and Start
- System chaincode

**See ： https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode4noah.html or Chaincode for Operators.DOC**

# Writing Your First Chaincode

We will explore chaincode through the eyes of an application developer. We'll present a asset-transfer chaincode sample walkthrough, and the purpose of each method in the Fabric Contract API.

我们将从应用程序开发人员的角度来研究链码。我们将展示一个资产转移链码示例演练，以及Fabric Contract API中每种方法的用途。

· 如果你是一个网络运营商，正在部署链码运行网络，访问部署智能合同到一个频道教程和织物链码生命周期概念主题。
· 本教程概述了Fabric合约API提供的高级API。
· 要了解有关使用Fabric合约API开发智能合约的更多信息，请访问智能合约处理主题

•If you are a network operator who is deploying a chaincode to running network, visit the Deploying a smart contract to a channel tutorial and the Fabric chaincode lifecycle concept topic.

•This tutorial provides an overview of the high level APIs provided by the Fabric Contract API.

•To learn more about developing smart contracts using the Fabric contract API, visit the Smart Contract Processing topic.

**See details in：https://hyperledger-fabric.readthedocs.io/en/latest/chaincode4ade.html or Writing Your First Chaincode.DOC**

# Deploying a production network

The process for deploying a Fabric network is complex and presumes an understanding of Public Key Infrastructure and managing distributed systems. If you are a smart contract or application developer, you should not need this level of expertise in deploying a production level Fabric network. However, you might need to be aware of how networks are deployed in order to develop effective smart contracts and applications.

部署Fabric网络的过程很复杂，需要了解公钥基础设施和管理分布式系统。如果您是一名智能合同或应用程序开发人员，那么在部署生产级Fabric网络时，您不应该需要这种级别的专业知识。但是，为了开发有效的智能合约和应用程序，您可能需要了解网络是如何部署的。

该指南将向您概述设置生产组件和生产网络的步骤：

The guide will give you an overview of the steps of setting up production components and a production network:

- **Step one: Decide on your network configuration**
- **Step two: Set up a cluster for your resources**
- **Step three: Set up your CAs**
- **Step four: Use the CA to create identities and MSPs**
- **Step five: Deploy nodes**
  - Create a peer
  - Create an ordering node

**See details in： https://hyperledger-fabric.readthedocs.io/en/latest/deployment_guide_overview.html or Deploying a production network.DOC**

# Using CouchDB

This tutorial will describe the steps required to use the CouchDB as the state database with Hyperledger Fabric.

- By now, you should be familiar with Fabric concepts and have explored some of the samples and tutorials.

本教程将描述使用CouchDB作为状态数据库和Hyperledger结构所需的步骤。
•到目前为止，您应该熟悉Fabric的概念，并探索了一些示例和教程。
本教程将带您完成以下步骤：

The tutorial will take you through the following steps:

- ➢ Enable CouchDB in Hyperledger Fabric
- ➢ Create an index
- ➢ Add the index to your chaincode folder
- ➢ Install and instantiate the Chaincode
- ➢ Query the CouchDB State Database
- ➢ Use best practices for queries and indexes
- ➢ Query the CouchDB State Database With Pagination
- ➢ Update an Index
- ➢ Delete an Index

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/couchdb_tutorial.html **or Using CouchDB.DOC**

# Videos

This collection（ledger-sprint3.MP4）contains developers demonstrating various v1 features and components such as: ledger, channels, gossip, SDK, chaincode, MSP, and more…

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/videos.html

# Contents

Prerequisites
Installations
Developing Applications
The scenario
Tutorials
Operations Guides
Commands Reference
Architecture Reference
Frequently Asked Questions

# Operations Guides

- Upgrading to the Newest Version of Fabric
- Setting up an ordering node
- Updating a Channel Configuration
- Membership Service Providers (MSP)
- Channel Configuration (configtx)
- Defining capability requirements
- Endorsement policies
- Pluggable transaction endorsement and validation
- Access Control Lists (ACL)
- MSP Implementation with Identity Mixer
- Identity Mixer MSP configuration generator (idemixgen)
- The Operations Service
- Metrics Reference
- Error handling
- Logging Control
- Securing Communication With Transport Layer Security (TLS)
- Configuring and operating a Raft ordering service
- Migrating from Kafka to Raft
- Bringing up a Kafka-based Ordering Service

**See details in：  https://hyperledger-fabric.readthedocs.io/en/latest/ops_guide.html or Operations Guides.DOC**

# Upgrading to the latest release

如果您熟悉Hyperledger Fabric的以前版本，那么您就会意识到，将节点和通道升级到Fabric的最新版本，在较高的层次上需要四个步骤。
1. 备份分类帐和MSPs。
2. 以滚动方式将订货程序二进制文件升级为最新的Fabric版本。
3.以滚动的方式将对等二进制文件升级到最新的Fabric版本。
4. 在可用的地方，将订购者系统通道和任何应用程序通道更新为最新的能力级别。
注意，有些版本在所有组中都有功能，而其他版本可能只有很少甚至根本没有新功能。
有关功能的更多信息，请查看通道功能。
要了解这些升级过程是如何完成的，请参阅这些教程:
升级您的组件。组件应该在更新任何功能之前升级到最新版本。
更新通道的能力级别。更新所有节点的版本后完成。
启用新的链码生命周期。在Fabric 2.0的新链码生命周期中添加组织特定的背书策略是必要的。
到目前为止，升级节点和增加通道的能力级别被认为是一个标准的Fabric流程，因此我们将不展示用于升级到最新版本的特定命令。类似地，fabric-samples repo中也没有脚本将样例网络从上一个版本升级到这个版本，就像以前的版本那样

If you're familiar with previous releases of Hyperledger Fabric, you're aware that upgrading the nodes and channels to the latest version of Fabric is, at a high level, a four step process.

1. Backup the ledger and MSPs.
2. Upgrade the orderer binaries in a rolling fashion to the latest Fabric version.
3. Upgrade the peer binaries in a rolling fashion to the latest Fabric version.
4. Update the orderer system channel and any application channels to the latest capability levels, where available.

Note that some releases will have capabilities in all groups while other releases may have few or even no new capabilities at all.

For more information about capabilities, check out Channel capabilities.

For a look at how these upgrade processes are accomplished, please consult these tutorials:

- Upgrading your components. Components should be upgraded to the latest version before updating any capabilities.
- Updating the capability level of a channel. Completed after updating the versions of all nodes.
- Enabling the new chaincode lifecycle. Necessary to add organization specific endorsement policies central to the new chaincode lifecycle for Fabric v2.0.

As the upgrading of nodes and increasing the capability levels of channels is by now considered a standard Fabric process, we will not show the specific commands for upgrading to the newest release. Similarly, there is no script in the fabric-samples repo that will upgrade a sample network from the previous release to this one, as there has been for previous releases.

**See details in： https://hyperledger-fabric.readthedocs.io/en/latest/upgrade.html or Upgrading to the latest release.DOC**

# Contents

Prerequisites

Installations

Developing Applications

The scenario

Tutorials

Operations Guides

Commands Reference

Architecture Reference

Frequently Asked Questions

# Commands Reference

- peer
- peer chaincode
- peer channel
- peer version
- peer logging
- peer node
- configtxgen
- configtxlator
- cryptogen
- Service Discovery CLI
- Fabric-CA Commands

**See details in：** https://hyperledger-fabric.readthedocs.io/en/release-1.4/command_ref.html or **Commands Reference**.**doc**

# Contents

Prerequisites
Installations
Developing Applications
The scenario
Tutorials
Operations Guides
Commands Reference
Architecture Reference
Frequently Asked Questions

# Architecture Reference

- Architecture Origins
- Transaction Flow
- Hyperledger Fabric CA's User Guide
- Hyperledger Fabric SDKs
- Service Discovery
- Channels
- CouchDB as the State Database
- Peer channel-based event services
- Private Data
- Read-Write set semantics
- Gossip data dissemination protocol

**See details in：** https://hyperledger-fabric.readthedocs.io/en/release-1.4/architecture.html or **Architecture Reference.doc**

# Contents

Prerequisites
Installations
Developing Applications
The scenario
Tutorials
Operations Guides
Commands Reference
Architecture Reference
Frequently Asked Questions

# Frequently Asked Questions

- ➢ Endorsement
- ➢ Endorsement architecture:
- ➢ Security & Access Control
- ➢ Application-side Programming Model
- ➢ Chaincode (Smart Contracts and Digital Assets)
- ➢ Differences in Most Recent Releases
- ➢ Ordering Service
- ➢ BFT

**See details in：** https://hyperledger-fabric.readthedocs.io/en/latest/Fabric-FAQ.html or **Frequently Asked Questions**.doc

# Contents

- ✓ Prerequisites
- ✓ Installations
- ✓ Developing Applications
- ✓ The scenario
- ✓ Tutorials
- ✓ Operations Guides
- ✓ Commands Reference
- ✓ Architecture Reference
- ✓ Frequently Asked Questions

- ✓ Key Concepts
- ✓ Developing an Application

# END !

धन्यवाद

Hindi Hindi

多謝

Traditional Chinese

ขอบคุณ

Thai

Спасибо

Russian

Gracias

Spanish

Thank You

English

شكرا

Arabic

Obrigado

Brazilian Portuguese

Grazie

Italia
n

多谢

Simplified Chinese

Danke

German

Merci

French

நன்றி

Tamil

ありがとうございました

Japanese

감사합니다

Korean