目录

[1 介绍 introduction 4](#_Toc11533)

[1.1 Hyperledger Fabric 4](#_Toc29702)

[1.2 Modularity 模块化 5](#_Toc6331)

[1.3 permissioned 许可 5](#_Toc19120)

[1.4 smart contracts 智能合约 6](#_Toc28098)

[1.5 A New Approach 新方式 7](#_Toc20006)

[1.6 Privacy and Confidentiality 隐私和机密性 7](#_Toc13025)

[1.7 Pluggable Consensus 可插拔性共识 9](#_Toc2800)

[1.8 Performance and Scalability 性能和可扩展性 9](#_Toc26674)

[1.9 Conclusion 结论 10](#_Toc30606)

[1.10 Acknowledgement 致谢 11](#_Toc3522)

[2 使用 getting started 11](#_Toc4415)

[2.1 prerequisites 预备 11](#_Toc15280)

[2.1.1 install cURL 12](#_Toc18453)

[2.1.2 Docker and Docker Compose 12](#_Toc16491)

[2.1.3 Go Programming Language 13](#_Toc31943)

[2.1.4 Node.js Runtime and NPM 14](#_Toc7631)

[2.1.5 Windows extras 15](#_Toc9681)

[2.2 install sample,Binries and Docker images 初始化简单的二进制Docker镜像 17](#_Toc17450)

[2.3 Hyperledger Fabric SDKs 20](#_Toc7915)

[2.4 Hyperledger Fabric CA 20](#_Toc8823)

[3 主要概念 key concepts 21](#_Toc31963)

[3.1 introduction 介绍 21](#_Toc7495)

[3.2 Hyperledger Fabric Functionalities 功能 21](#_Toc20779)

[3.3 Hyperledger Fabric Model 模型 21](#_Toc19000)

[3.4 Hyperledger Fabric Network 网络 21](#_Toc25682)

[3.5 Identity 身份 21](#_Toc31727)

[3.6 Membership 成员 21](#_Toc25160)

[3.7 Peers 节点 21](#_Toc19167)

[3.8 Private data 私有数据 22](#_Toc5283)

[3.9 Ledger 账本 22](#_Toc26993)

[3.10 Use Cases 用例 22](#_Toc24233)

[4 教程 tutorials 22](#_Toc17991)

[4.1 Writing Your First Application 第一个应用 22](#_Toc4871)

[4.2 Building Your First Network 第一个网络 22](#_Toc21933)

[4.3 Adding an Org to a Channel 子链中增加一个组织 22](#_Toc14334)

[4.4 Upgrading Your Network Components 升级网络组件 23](#_Toc25458)

[4.5 Using Private Data in Fabric 使用链码私有数据 23](#_Toc13901)

[4.6 Chaincode Tutorials 链码实例 23](#_Toc6344)

[4.7 Chaincode for Developers 开发链码 23](#_Toc4131)

[4.8 Chaincode for Operators 运营链码 23](#_Toc32216)

[4.9 System Chaincode Plugins 系统链码插件 23](#_Toc26841)

[4.10 Using CouchDB 使用Videos 23](#_Toc15653)

[4.11 Videos 视频 23](#_Toc11079)

[5 运营指导 operation guides 24](#_Toc24319)

[5.1 Upgrading to the Newest Version of Fabric 24](#_Toc19437)

[5.2 Updating a Channel Configuration 24](#_Toc9617)

[5.3 Membership Service Providers (MSP) 24](#_Toc19789)

[5.4 Channel Configuration (configtx) 24](#_Toc13900)

[5.5 Endorsement policies 24](#_Toc8231)

[5.6 Pluggable transaction endorsement and validation 24](#_Toc25534)

[5.7 Access Control Lists (ACL) 24](#_Toc1650)

[5.8 Error handling 24](#_Toc17177)

[5.9 Logging Control 24](#_Toc5963)

[5.10 Securing Communication With Transport Layer Security (TLS) 25](#_Toc21235)

[5.11 Bringing up a Kafka-based Ordering Service 25](#_Toc13105)

[6 相关命令 commands reference 25](#_Toc25864)

[6.1 peer 25](#_Toc29507)

[6.2 peer chaincode 25](#_Toc19201)

[6.3 peer channel 25](#_Toc13156)

[6.4 peer version 25](#_Toc1686)

[6.5 peer logging 25](#_Toc5732)

[6.6 peer node 25](#_Toc32633)

[6.7 configtxgen 26](#_Toc7339)

[6.8 configtxlator 26](#_Toc5619)

[6.9 cryptogen 26](#_Toc26867)

[6.10 Service Discovery Command Line Interface (discover) 26](#_Toc24348)

[6.11 Fabric-CA Commands 26](#_Toc27665)

[7 相关架构 Architcture reference 26](#_Toc17)

[7.1 Architecture Explained 26](#_Toc14312)

[7.2 Transaction Flow 26](#_Toc1939)

[7.3 Hyperledger Fabric CA's User Guide 26](#_Toc31469)

[7.4 Hyperledger Fabric SDKs 27](#_Toc25082)

[7.5 Service Discovery 27](#_Toc28913)

[7.6 Channels 27](#_Toc8545)

[7.7 Capability Requirements 27](#_Toc15111)

[7.8 CouchDB as the State Database 27](#_Toc25704)

[7.9 Peer channel-based event services 27](#_Toc24506)

[7.10 Private Data 27](#_Toc14357)

[7.11 Read-Write set semantics 27](#_Toc32252)

[7.12 Gossip data dissemination protocol 27](#_Toc30095)

[8 常见问题 frequently asked questions 28](#_Toc20588)

[9 代码贡献 contributions welcome 28](#_Toc25284)

[10 术语 glossary 28](#_Toc19428)

[10.1 Introduction 28](#_Toc21822)

[10.2 Hyperledger Fabric Functionalities 28](#_Toc15220)

[10.3 Hyperledger Fabric Model 28](#_Toc20662)

[10.4 Hyperledger Fabric Network 28](#_Toc3377)

[10.5 Identity 28](#_Toc10022)

[10.6 Membership 28](#_Toc24197)

[10.7 Peers 29](#_Toc3680)

[10.8 Private data 29](#_Toc3034)

[10.9 Ledger 29](#_Toc26336)

[10.10 Use Cases 29](#_Toc5778)

[11 其他 29](#_Toc4026)

[11.1 术语 glossary 29](#_Toc7608)

[11.2 提问 still have questions 29](#_Toc4307)

[11.3 各版本简介 29](#_Toc1010)

[11.3.1 1.0 版本介绍 29](#_Toc6707)

[11.3.2 1.1 版本介绍 30](#_Toc18886)

[11.3.3 1.2 版本介绍 30](#_Toc24703)

[11.3.4 1.3 版本介绍 31](#_Toc30377)

[11.4 Fabric CA 31](#_Toc1302)

[词汇 31](#_Toc1856)

<https://hyperledger-fabric.readthedocs.io/en/release-1.2/whatis.html>

# 1 介绍 introduction

The first and most widely recognized application of blockchain is the [Bitcoin](https://en.wikipedia.org/wiki/Bitcoin) cryptocurrency.Bitcoin and Ethereum fall into a class of blockchain that we would classify as public permissionless blockchain technology. Basically, these are public networks, open to anyone, where participants interact anonymously.

For enterprise use, we need to consider the following requirements:

1 Participants must be identified/identifiable （安全）

2 Networks need to be permissioned （安全）

3 High transaction throughput performance （性能）

4 Low latency of transaction confirmation （性能）

5 Privacy and confidentiality of transactions and data pertaining to business transactions （安全）

## 1.1 Hyperledger Fabric

Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms.

--针对企业应用的分布式账本技术

Hyperledger Fabric project by a diverse set of maintainers from multiple organizations. It has a development community that has grown to over 35 organizations and nearly 200 developers since its earliest commits.

--目前有35个组织和200多名开发人员提交代码

Fabric has a highly modular and configurable architecture

--高度模块化可配置化

Fabric is the first distributed ledger platform to support smart contracts authored in general-purpose programming languages such as Java, Go and Node.js, rather than constrained domain-specific languages (DSL).

--第一个用通用开发语言支持的合约，而不需特定领域开发语言

its support for pluggable consensus protocols that enable the platform to be more effectively customized to fit particular use cases and trust models.

--支持可插拔的协议 更有效的客户定制化

Fabric can leverage consensus protocols that do not require a native cryptocurrency to incent costly mining or to fuel smart contract execution.

--fabric无需加密货币及合约运行的燃料

## 1.2 Modularity 模块化

the platform has been designed at its core to be configured to meet the diversity of enterprise use case requirements.

--面向企业工业需求的可配置化设计

Hyperledger Fabric can be configured in multiple ways to satisfy the diverse solution requirements for multiple industry use cases.、

--Fabric能够通过配置满足不同的工业需求

Fabric is comprised of the following modular components:

--fabric包括了一系列模块化组件

1 A pluggable ordering service establishes consensus on the order of transactions and then broadcasts blocks to peers.

插拔式的排序服务确保共识有序并广播其他peer节点

2 A pluggable membership service provider is responsible for associating entities in the network with cryptographic identities.

插拔式的用户管理服务通过加密身份在网络下不同企业联系在一起

3 An optional peer-to-peer gossip service disseminates the blocks output by ordering service to other peers.

可选的gossip点对点的通讯服务 用于节点之间的信息通信

4 Smart contracts (“chaincode”) run within a container environment (e.g. Docker) for isolation. They can be written in standard programming languages but do not have direct access to the ledger state.

合约单独运行在一个docker容器中，相互之间隔离。可能使用标准的编程语言，但不能访问区块状态

5 The ledger can be configured to support a variety of DBMSs.

账本存储支持不同的数据存储管理系统

6 A pluggable endorsement and validation policy enforcement that can be independently configured per application.

一个可插拔的背书策略和策略确认可以在每个应用独立配置

## 1.3 permissioned 许可

In a permissionless blockchain, every participant is anonymous.

In order to mitigate this absence of trust, permissionless blockchains typically employ a “mined” native cryptocurrency or transaction fees to provide economic incentive to offset the extraordinary costs of participating in a form of byzantine fault tolerant consensus based on “proof of work” (PoW).

为了缓解在匿名情况下的信任问题，有了挖矿和gas ，为了抵消拜占庭容错问题有了激励

Permissioned blockchains, on the other hand, operate a blockchain amongst a set of known, identified and often vetted participants operating under a governance model that yields a certain degree of trust.

许可的区块链存在另外的问题，需要运维一系列资料、提供证书并验证参与者的合法性

Additionally, in such a permissioned context, the risk of a participant intentionally introducing malicious code through a smart contract is diminished

另外 在授信环境下，通过合约引入恶意代码的风险减小。

First, the participants are known to one another and all actions

首先参与者都知道对方及行为

submitting application transactions, modifying the configuration of the network or deploying a smart contract are recorded on the blockchain following an endorsement policy that was established for the network and relevant transaction type.

其次 不论提交事务 还是部署合约、修改配置，都要通过网络经过背书策略，

## 1.4 smart contracts 智能合约

A smart contract, or what Fabric calls “chaincode”，。as a trusted distributed application that gains its security/trust from the blockchain

小的智能合约，fabric称为链码。作为一个可信任的分布式应用。

The order-execute architecture can be found in virtually all existing blockchain systems, ranging from public/permissionless platforms such as Ethereum (with PoW-based consensus) to permissioned platforms such as Tendermint, Chain, and Quorum.

共识服务存在于已有的区块链系统中，不论是公开的平台 （如以太坊） 还有私有的平台 如Tendermint、Chain、Quorum

Smart contracts executing in a blockchain that operates with the order-execute architecture must be deterministic，otherwise, consensus might never be reached;

合约的执行在链码上执行必须是没有二义性，否则合约可以不执行

many platforms require that the smart contracts be written in a non-standard, or domain-specific language (such as Solidity) so that non-deterministic operations can be eliminated. This hinders wide-spread adoption because it requires developers writing smart contracts to learn a new language and may lead to programming errors.

合约不需要学习一门新语言，比如solidity，也不会因此引入语法错误。

since all transactions are executed sequentially by all nodes, performance and scale is limited.

所有事务在所有节点必须是有序的，性能和规模因此被限制。

## 1.5 A New Approach 新方式

the transaction flow into three steps:

一个交易遵从以下三步

1 execute a transaction and check its correctness, thereby endorsing it,

执行一个交易 并且确认他的正确性 然后背书他

2 order transactions via a (pluggable) consensus protocol, and

通过共识服务，进行排序交易

3 validate transactions against an application-specific endorsement policy before committing them to the ledger

提交到账本之前，先验证特殊应用的背书策略。

In Fabric, an application-specific endorsement policy specifies which peer nodes, or how many of them, need to vouch for the correct execution of a given smart contract.

在fabric中，不论是全部节点还是特定的指定节点，应用的特定背书策略都要确保执行的正确性。

Thus, each transaction need only be executed (endorsed) by the subset of the peer nodes necessary to satisfy the transaction’s endorsement policy.

因此，每个事务需要仅仅系列认可的背书节点中安全的执行。

This allows for parallel execution increasing overall performance and scale of the system.

他允许并行执行增加了整体性能和系统扩展性。

This first phase also eliminates any non-determinism, as inconsistent results can be filtered out before ordering.

第一个阶段同样消除了任何不确定性，同样不一致性结果在排序之前也被过滤掉。

Because we have eliminated non-determinism, Fabric is the first blockchain technology that enables use of standard programming languages. In the 1.1.0 release, smart contracts can be written in either Go or Node.js, while there are plans to support other popular languages including Java in subsequent releases

因为我们消除了不确定性，Fabric是第一个支持标准编程语言的区块链技术，在1.1.0 release版本中，智能合约可以用Go或node.js编写，有计划支持更多流行的编程语言支持在后续版本中。

## 1.6 Privacy and Confidentiality 隐私和机密性

As we have discussed, in a public, permissionless blockchain network that leverages PoW for its consensus model, transactions are executed on every node.

正如我们所讨论的，在一个公共场所，没有许可的区块链账本利用PoW实现共识，交易在每个节点上执行。

This means that neither can there be confidentiality of the contracts themselves, nor of the transaction data that they process.

这意味着既不能给合约本身加密，也不能给交易数据加密

Every transaction, and the code that implements it, is visible to every node in the network.

针对每笔交易，网络中的每个节点都是可见的。

In this case, we have traded confidentiality of contract and data for byzantine fault tolerant consensus delivered by PoW.

在这种情况下，我们用合约和数据的安全性换取了使用PoW容错替代拜占庭的容错性。

This lack of confidentiality can be problematic for many business/enterprise use cases.

对很多业务/企业来说，缺乏保密性是一个大问题

For example, in a network of supply-chain partners, some consumers might be given preferred rates as a means of either solidifying a relationship, or promoting additional sales.

比如，在一个供应链网络中，一些客户可能需要给予优惠价格，作为巩固关系或促进额外销售。

If every participant can see every contract and transaction, it becomes impossible to maintain such business relationships in a completely transparent network – everyone will want the preferred rates!

如果每个合作伙伴都能看到合约和交易，在一个透明的网络下，大家都想这种折扣，这类关系的维护就变得不可能。

As a second example, consider the securities industry, where a trader building a position (or disposing of one) would not want her competitors to know of this, or else they will seek to get in on the game, weakening the trader’s gambit.

在举一个例子，在证券行业，交易员构建一个头寸买卖测策略，不希望竞争对手知道，否则他们会想办法参与进来，最终会削弱交易结果。

In order to address the lack of privacy and confidentiality for purposes of delivering on enterprise use case requirements, blockchain platforms have adopted a variety of approaches.

为了解决为交付企业用例需求而缺乏隐私和机密性问题，区块链平台采用了多种方法

All have their trade-offs.

所有人都有他们的取舍。

Encrypting data is one approach to providing confidentiality;

加密数据是提供机密性的一种方法

however, in a permissionless network leveraging PoW for its consensus, the encrypted data is sitting on every node.

然后，在没有信任的网络环境中，采用PoW达成共识，每个节点存储加密数据。

Given enough time and computational resource, the encryption could be broken.

只要时间和资源下，加密可能被破解。

For many enterprise use cases, the risk that their information could become compromised is unacceptable.

在很多企业，信息被泄露的风险是不能接受的。

In a permissioned context that can leverage alternate forms of consensus, one might explore approaches that restrict the distribution of confidential information exclusively to authorized nodes.

在一个授权的环境下，可以利用其它形式的协商一致意见，可以探索将机密信息完全分配给授权节点的方法。

In a permissioned context that can leverage alternate forms of consensus, one might explore approaches that restrict the distribution of confidential information exclusively to authorized nodes.

在一个授权的环境下，可以利用其它形式的协商一致意见，可以探索将机密信息完全分配给授权节点的方法。

## 1.7 Pluggable Consensus 可插拔性共识

The ordering of transactions is delegated to a modular component for consensus that is logically decoupled from the peers that execute transactions and maintain the ledger.

交易排序工作委托给了共识组件，该组件把事务执行和账本顺序维护进行逻辑分离。

Specifically, the ordering service. Since consensus is modular, its implementation can be tailored to the trust assumption of a particular deployment or solution.

具体而言，共识服务因为是模块化的，可能根据部署和解决方案在信任场景假设情况来进行定制化。

This modular architecture allows the platform to rely on well-established toolkits for CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) ordering.

这种模块化架构允许平台依赖成熟的工具包进行CFT(崩溃容错)或BFT(拜占庭容错)的排序

In the currently available releases, Fabric offers a CFT ordering service implemented with Kafka and Zookeeper.

在当前已经发布的版本中，fabric提供了用kafka和zookeepr实现的CFT交易排序服务。

In subsequent releases, Fabric will deliver a Raft consensus ordering service implemented with etcd/Raft and a fully decentralized BFT ordering service.

在接下来的版本中，fabric将发布基于Raft的排序服务使用etcd/Raft算法 和 完全分散的BFT排序服务

Note also that these are not mutually exclusive. A Fabric network can have multiple ordering services supporting different applications or application requirements.

值得注意的是这些排序服务之间不是相互排斥的，一个fabric网络支持不同的排序服务针对不同的业务应用

## 1.8 Performance and Scalability 性能和可扩展性

Performance of a blockchain platform can be affected by many variables such as transaction size, block size, network size, as well as limits of the hardware, etc.

区块链平台的性能受到很多因素的影响，比如事务内容大小、区块大小、磁盘的限制等。

The Hyperledger community is currently developing a draft set of measures within the Performance and Scale working group, along with a corresponding implementation of a benchmarking framework called Hyperledger Caliper.

Hyperledger社区工作小组研发了一系列测量性能和扩展性的草案，以及称之为Hyperledger分类卡的基准实现。

While that work continues to be developed and should be seen as a definitive measure of blockchain platform performance and scale characteristics,

虽然这项研发工作还在进行，不过应该被作为区块链衡量性能和扩展性的标准。

a team from IBM Research has published a peer reviewed paper that evaluated the architecture and performance of Hyperledger Fabric.

IBM的研究团队发布了篇关于区块链节点性能和扩展性评估方面的论文，

The paper offers an in-depth discussion of the architecture of Fabric and then reports on the team’s performance evaluation of the platform using a preliminary release of Hyperledger Fabric v1.1.

该论文深度讨论了fabric体系结构，然后团队对fabric1.1所做了一个初步的评测报告。

The benchmarking efforts that the research team did yielded a significant number of performance improvements for the Fabric v1.1.0 release that more than doubled the overall performance of the platform from the v1.0.0 release levels.

通过基准测试，研究团队已经促从V1.0.0 性能到v1.1.0 整体性能提升了一倍。

## 1.9 Conclusion 结论

Any serious evaluation of blockchain platforms should include Hyperledger Fabric in its short list.

任何有价值的功能都应该包括在区块链短功能列表清单中。

Combined, the differentiating capabilities of Fabric make it a highly scalable system for permissioned blockchains supporting flexible trust assumptions

总而言之，区块链不同的能力使得fabric称为一个高可扩展的持久化区块链系统。

that enable the platform to support a wide range of industry use cases ranging from government, to finance, to supply-chain logistics, to healthcare and so much more.

使得平台更大范围的支持包括政府、金融、供应链物流、医疗保健等

More importantly, Hyperledger Fabric is the most active of the (currently) ten Hyperledger projects. The community building around the platform is growing steadily,

更重要的，fabric 有最多的活跃活跃项目，围绕平台的社区人员稳定增长，

and the innovation delivered with each successive release far out-paces any of the other enterprise blockchain platforms.

并且每次版本的发版 创新元素远远超过其他企业级区块链平台。

Hyperledger Fabric, being a permissioned platform, enables confidentiality through its channel architecture.

Hyperledger Fabric 作为一个授信平台，通过子链架构来确保其机密性。

Basically, participants on a Fabric network can establish a “channel” between the subset of participants that should be granted visibility to a particular set of transactions.

基本上，Fabric网络上的参与者可以在参与者之间建立一个"通道"，通过通道授予事务可见性。

Think of this as a network overlay.

这种授权是整个网络覆盖的。

Thus, only those nodes that participate in a channel have access to the smart contract (chaincode) and data transacted, preserving the privacy and confidentiality of both.

因此，只有那些参与通道的节点才能访问智能契约(链码)和数据事务，从而保护两者的隐私和机密性。

To improve upon its privacy and confidentiality capabilities, Fabric has added support for private data and is working on zero knowledge proofs (ZKP) available in the future. More on this as it becomes available.

为了改进隐私和保密能力，fabric将来的版本中会增加ZKP(零知识证明)的支持。这一点将变得更可用。

## 1.10 Acknowledgement 致谢

“Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”

https://arxiv.org/abs/1801.10228v1

# 2 使用 getting started

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.

在我们开始之间，需要准备开发或运用区块链的平台上,初始化准备工作。

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

一旦安装了先决条件，就可以下载安装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.

为了Fabric二进制文件开发的安装，我们提供了一个初始化脚本来构建简单二进制和docker镜像系统，

The script also will download the Docker images to your local registry.

这个脚本还将下载docker镜像到你的本地注册表

## 2.1 prerequisites 预备

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.

在我们开始之间，需要准备开发或运用区块链的平台上,初始化准备工作。

### 2.1.1 install cURL

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.

如果还没有下载环境或者通过命令做了错误的下载，使用cURL 工具下载最新版本

cURL 下载地址

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

note:注意

If you’re on Windows please see the specific note on Windows extras below.

如果是windos环境，请查看下面windows extras说明

### 2.1.2 Docker and Docker Compose

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

你需要按下列操作在Fabric区块链开发或运营平台上进行安装

1 MacOSX, \*nix, or Windows 10: Docker Docker version 17.06.2-ce or greater is required.

mac操作系统 或 win10 Docker需要在17.06.2-ce 以上版本

2 Older versions of Windows: Docker Toolbox - again, Docker version Docker 17.06.2-ce or greater is required.

windows更早的版本 ：Docker 工具版本一样 Docker需要在17.06.2-ce 以上版本

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

可以通过下列终端提示命令，确认docker的版本号

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

docker --version

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Note：注

Installing Docker for Mac or Windows, or Docker Toolbox will also install Docker Compose.

在Mac或Windows安装Docker，Docker Toolbox 一样可以安装Docker Compose。

If you already had Docker installed, you should check that you have Docker Compose version 1.14.0 or greater installed.

如果已经安装了Docker,你应该确认Docker compass的版本应该在1.14.0以上。

If not, we recommend that you install a more recent version of Docker.

如果没有，我们建议您安装Docker的最新版本。

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

使用下面的终端命令确认Docker Compose版本

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

docker-compose --version

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

### 2.1.3 Go Programming Language

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

Fabric的很多组件都使用的Go语言

Go version 1.10.x is required.

--Go版本需要在1.10.x

Given that we will be writing chaincode programs in Go, there are two environment variables you will need to set properly;

基于我们要使用Go语言编写链码，有两个环节变量需要设置属性。

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.

可以通过设置放在适启动位置永久保留，如果使用的linux系统，在 ~/.bashrc 文件中设置。

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

首先，你必须设置环境变量GOPATH，用来作为下载fabric代码基础地址 ，比如：

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

export GOPATH=$HOME/go

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Note 注

You must set the GOPATH variable

你必须设置GOPATH变量

Even though, in Linux, Go’s GOPATH variable can be a colon-separated list of directories, and will use a default value of $HOME/go if it is unset,

即使在Linux环境下，Go变量可以采用系列分号分割，如果没有设置并将使用默认变量$HOME/go。

the current Fabric build framework still requires you to set and export that variable, and it must contain only the single directory name for your Go workspace.

当前Fabric构建框架仍需要设置并导入该变量，而且他必须包括Go工作区的单个目录名。

(This restriction might be removed in a future release.)

这个限制可能在将来版本中删除。

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:

其次，您应该扩展命令搜索路径，以包括Go bin 目录，linux采用下面例子

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

export PATH=$PATH:$GOPATH/bin

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

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.

虽然在新安装Go工作空间这个目录可能不存在，稍后Fabric构建系统将放入一些Go可执行文件，构建系统的其他部分将使用这些文件。

So even if you currently have no such directory yet, extend your shell search path as above.

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shell搜索路径。

If you will be developing applications for Hyperledger Fabric leveraging the Hyperledger Fabric SDK for Node.js, you will need to have version 8.9.x of Node.js installed.

如果要使用Farbric开发应用程序，使用Node.js作为fabric sdk,需要安装node.js 8.9.x以上版本。

### 2.1.4 Node.js Runtime and NPM

If you will be developing applications for Hyperledger Fabric leveraging the Hyperledger Fabric SDK for Node.js, you will need to have version 8.9.x of Node.js installed.

如果要使用Farbric开发应用程序，使用Node.js作为fabric sdk,需要安装node.js 8.9.x以上版本。

note:注

Node.js version 9.x is not supported at this time.

Node.js 9.x版本目前不支持。

Node.js - version 8.9.x or greater

采用Node.js 8.9.x 或更早

note:注

Installing Node.js will also install NPM, however it is recommended that you confirm the version of NPM installed. You can upgrade the npm tool with the following command:

安装node.js还需要安装NPM,需确认一下NPM安装的版本，可以通过下面命令更新npm 工具包

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

npm install npm@5.6.0 -g

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

#### 2.1.4.1 python

note 注

The following applies to Ubuntu 16.04 users only.

下列应用仅在Ubuntu 16.04 版本中使用

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.

默认情况下，Ubuntu 16.04 自带python3.5.1二进制安装。Fabric node.js sdk需要Python2.7的迭代才能成功完成npm安装操作。

Retrieve the 2.7 version with the following command:

使用以下命令检索2.7版本

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

sudo apt-get install python

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Check your version(s):

确认你的版本

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

python --version

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

### 2.1.5 Windows extras

If you are developing on Windows 7, you will want to work within the Docker Quickstart Terminal which uses Git Bash and provides a better alternative to the built-in Windows shell.

如果你在win7下开发，如果你希望在Docker Quickstart终端中运行，该终端使用Git Bash，并提供比内置Windows shell更好选择。

However experience has shown this to be a poor development environment with limited functionality.

然而，经验表明这是一个功能有限的开发环境。

It is suitable to run Docker based scenarios, such as Getting Started, but you may have difficulties with operations involving the make and docker commands.

它适合运行基于Docker的场景。例如在开始使用的时候，在make和制作docker命令遇到各种困难。

On Windows 10 you should use the native Docker distribution and you may use the Windows PowerShell.

如果再window10上运行，你需要使用本机Docker发行版，并且可以使用windows PowerShell.

However, for the binaries command to succeed you will still need to have the uname command available.

然而，要使二进制命令成功运行，还需要使用uname命令。

You can get it as part of Git but beware that only the 64bit version is supported.

你可以将其作为Git的一部分获得，但是注意只支持64位版本。

Before running any git clone commands, run the following commands:

在运行git命令以前，需运行以下命令

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

git config --global core.autocrlf false

git config --global core.longpaths true

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

You can check the setting of these parameters with the following commands:

可以使用下列命令确认设置的参数

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

git config --get core.autocrlf

git config --get core.longpaths

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

These need to be false and true respectively.

需要分别分会false和true返回值。

Note：注

If you have questions not addressed by this documentation, or run into issues with any of the tutorials, please visit the Still Have Questions? page for some tips on where to find additional help.

如果在文档中没有提到的问题，或者教程中任何问题，请到下面地址查看，里面有一些额外的提示。

The curl command that comes with Git and Docker Toolbox is old and does not handle properly the redirect used in Getting Started.

Git和Docker工具箱附带的curl命令很旧，不能正确地处理开始时使用的重定向。

Make sure you install and use a newer version from the cURL downloads page

确保安装并使用cURL下载页面的新版本。

For Node.js you also need the necessary Visual Studio C++ Build Tools which are freely available and can be installed with the following command:

为Node.js安装还需要安装Visual Studio C++构建，这些工具都是免费的，可以通过下面命令安装

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

npm install --global windows-build-tools

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

See the NPM windows-build-tools page for more details.

可以从NPM页面查看更多构建细节

https://www.npmjs.com/package/windows-build-tools

Once this is done, you should also install the NPM GRPC module with the following command:

一旦执行完，通过以下命令确认安装NPG GRPC模块

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

npm install --global grpc

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Your environment should now be ready to go through the Getting Started samples and tutorials.

你现在环境现在应该已经准备好进行入门示例和教程。

## 2.2 install sample,Binries and Docker images 初始化简单的二进制Docker镜像

While we work on developing real installers for the Hyperledger Fabric binaries, we provide a script that will download and install samples and binaries to your system.

当下开始fabric二进制文件安装的的话，我们提供了一个脚本用来下载和安装二进制包。

We think that you’ll find the sample applications installed useful to learn more about the capabilities and operations of Hyperledger Fabric.

我们认为你通过简单的运用可以学习更多关于fabric的能力和操作。

|  |
| --- |
| **Note 注** |
| If you are running on Windows you will want to make use of the Docker Quickstart Terminal for the upcoming terminal commands. Please visit the Prerequisites if you haven’t previously installed it.  如果运行在windos上，你想使用Docker Quickstart 终端来执行终端命令，如果之前还没有安装，请先访问先决条件。  If you are running on Windows you will want to make use of the Docker Quickstart Terminal for the upcoming terminal commands. Please visit the Prerequisites if you haven’t previously installed it.  如果运行在windos上，你想使用Docker Quickstart 终端来执行终端命令，如果之前还没有安装，请先访问先决条件。  If you are using Docker Toolbox on Windows 7 or macOS, you will need to use a location under C:\Users (Windows 7) or /Users (macOS) when installing and running the samples.  如果你在win7或macOs下使用docker,你将需要在c:\Users 或 /Users 目录下运行示例  If you are using Docker for Mac, you will need to use a location under /Users, /Volumes, /private, or /tmp. To use a different location, please consult the Docker documentation for file sharing.  如果要在Mac下使用Docker，你需要使用以下目录之一 /Users, /Volumes, /private, or /tmp。如果需要使用不同的目录，请参阅Docker文档共享文件。  If you are using Docker for Windows, please consult the Docker documentation for shared drives and use a location under one of the shared drives.  如果想在windos下使用docker,请参阅Docker文档共享驱动并在共享驱动下使用一个本地目录。 |

Determine a location on your machine where you want to place the fabric-samples repository and enter that directory in a terminal window.

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。

The command that follows will perform the following steps:

下面命令将执行以下步骤：

1 If needed, clone the hyperledger/fabric-samples repository

如果需要，克隆fabric-samples到仓库

2 Checkout the appropriate version tag

检出指定版本

3 Install the Hyperledger Fabric platform-specific binaries and config files for the version specified into the /bin and /config directories of fabric-samples

在fabric-samples的/bin/和/config目录中安装指定版本的超级账本Fabric平台专用二进制文件和配置文件

4 Download the Hyperledger Fabric docker images for the version specified

下载指定版本的fabric镜像

Once you are ready, and in the directory into which you will install the Fabric Samples and binaries, go ahead and execute the following command:

准备完成之后，在准备安装Fabric 实例、二进制、Go环境的目录下，先执行下面命令

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.1

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

|  |
| --- |
| **Note 注** |
| If you get an error running the above curl command, you may have too old a version of curl that does not handle redirects or an unsupported environment.  如果crul命令将运行错误，你可能是因为curl版本太老没有处理重定向或一个不支持的环境。  Please visit the Prerequisites page for additional information on where to find the latest version of curl and get the right environment.  请访问准备的额外信息页，那里有最新的crul版本并获取正确的环境变量  Alternately, you can substitute the un-shortened URL:  或者，你可以用没有缩写的地址替换  https://raw.githubusercontent.com/hyperledger/fabric/master/scripts/bootstrap.sh |
| You can use the command above for any published version of Hyperledger Fabric. Simply replace 1.2.1 with the version identifier of the version you wish to install.  你可以使用以上命令发行已发布的fabric所有版本，简单把1.2.1 版本号标识替换自己想要的即可。 |

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.

上面的命令将下载和执行脚本，脚本会下载平台所需的二进制包，这些二进制文件来构建你的fabric网络，克隆下面文件放入repo中。

It retrieves the following platform-specific binaries:

它检索下列二进制平台文件

configtxgen,

configtxlator,

cryptogen,

idemixgen

orderer,

peer, and

fabric-ca-client

and places them in the bin sub-directory of the current working directory.

并且替换他们在当前目录的子目录下。

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.:

你如果将路径增加到环境变量中，这样就可以不指定路径的情况下获取这些二进制功能。例如

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

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’.

最后，脚本将从Docker Hub下载fabric镜像到本地注册中心，并标记为'latest'

The script lists out the Docker images installed upon conclusion.

脚本列出了再结束安装的Docker映像。

Look at the names for each image;

看看每个镜像的名称

these are the components that will ultimately comprise our Hyperledger Fabric network.

上面这些组件将组成fabric网络的基础

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”.

你将可以注意到你有两个相同镜像实例，一个标记为“amd64-1.x.x” ，另一个标记为“amd64-1.x.x”

Prior to 1.2.0, the image being downloaded was determined by uname -m and showed as “x86\_64-1.x.x”.

在1.2.0之前，已下载的镜像使用uname -m确定，显示为“x86\_64-1.x.x”.

|  |
| --- |
| note注 |
| On different architectures, the x86\_64/amd64 would be replaced with the string identifying your architecture.  在不同的架构下，x86\_64/amd64 将被替换为标识体系结构的字符串。 |
| If you have questions not addressed by this documentation, or run into issues with any of the tutorials, please visit the Still Have Questions? page for some tips on where to find additional help.  如果遇到任何文档中没有提到的问题，或者运行实例中遇到的问题，请访问遗留问题网，一些tips在一些额外帮助页找到。 |

## 2.3 Hyperledger Fabric SDKs

Hyperledger Fabric offers a number of SDKs to support various programming languages. There are two officially released SDKs for Node.js and Java:

Hyperledger Fabric提供了很多SDK支持各种编程语言，下面提供了二种sdk node.js 和java

1 Hyperledger Fabric Node SDK and Node SDK documentation.

<https://github.com/hyperledger/fabric-sdk-node>

2 Hyperledger Fabric Java SDK.

<https://github.com/hyperledger/fabric-sdk-java>

In addition, there are three more SDKs that have not yet been officially released (for Python, Go and REST), but they are still available for downloading and testing:

另外，还有三种SDK(python\Go\rest)没有发布,不过也可以下载测试

1 Hyperledger Fabric Python SDK.

<https://github.com/hyperledger/fabric-sdk-py>

2 Hyperledger Fabric Go SDK.

<https://github.com/hyperledger/fabric-sdk-go>

3 Hyperledger Fabric REST SDK.

<https://github.com/hyperledger/fabric-sdk-rest>

## 2.4 Hyperledger Fabric CA

Hyperledger Fabric provides an optional certificate authority service that you may choose to use to generate the certificates and key material to configure and manage identity in your blockchain network.

Hyperledger Fabric提供了一个可选的认证授权服务，可以用来生成证书和关键材料用来管理区块链网络中的身份认证。

However, any CA that can generate ECDSA certificates may be used.

然而，任何CA都可以ECDSA 证书

# 3 主要概念 key concepts

## 3.1 introduction 介绍

## 3.2 Hyperledger Fabric Functionalities 功能

## 3.3 Hyperledger Fabric Model 模型

## 3.4 Hyperledger Fabric Network 网络

## 3.5 Identity 身份

## 3.6 Membership 成员

## 3.7 Peers 节点

## 3.8 Private data 私有数据

## 3.9 Ledger 账本

## 3.10 Use Cases 用例

# 4 教程 tutorials

## 4.1 Writing Your First Application 第一个应用

## 4.2 Building Your First Network 第一个网络

## 4.3 Adding an Org to a Channel 子链中增加一个组织

## 4.4 Upgrading Your Network Components 升级网络组件

## 4.5 Using Private Data in Fabric 使用链码私有数据

## 4.6 Chaincode Tutorials 链码实例

## 4.7 Chaincode for Developers 开发链码

## 4.8 Chaincode for Operators 运营链码

## 4.9 System Chaincode Plugins 系统链码插件

## 4.10 Using CouchDB 使用Videos

## 4.11 Videos 视频

# 5 运营指导 operation guides

## 5.1 Upgrading to the Newest Version of Fabric

## 5.2 Updating a Channel Configuration

## 5.3 Membership Service Providers (MSP)

## 5.4 Channel Configuration (configtx)

## 5.5 Endorsement policies

## 5.6 Pluggable transaction endorsement and validation

## 5.7 Access Control Lists (ACL)

## 5.8 Error handling

## 5.9 Logging Control

## 5.10 Securing Communication With Transport Layer Security (TLS)

## 5.11 Bringing up a Kafka-based Ordering Service

# 6 相关命令 commands reference

## 6.1 peer

## 6.2 peer chaincode

## 6.3 peer channel

## 6.4 peer version

## 6.5 peer logging

## 6.6 peer node

## 6.7 configtxgen

## 6.8 configtxlator

## 6.9 cryptogen

## 6.10 Service Discovery Command Line Interface (discover)

## 6.11 Fabric-CA Commands

# 7 相关架构 Architcture reference

## 7.1 Architecture Explained

## 7.2 Transaction Flow

## 7.3 Hyperledger Fabric CA's User Guide

## 7.4 Hyperledger Fabric SDKs

## 7.5 Service Discovery

## 7.6 Channels

## 7.7 Capability Requirements

## 7.8 CouchDB as the State Database

## 7.9 Peer channel-based event services

## 7.10 Private Data

## 7.11 Read-Write set semantics

## 7.12 Gossip data dissemination protocol

# 8 常见问题 frequently asked questions

# 9 代码贡献 contributions welcome

见fabric 贡献文档

# 10 术语 glossary

# 11 其他

## 11.1 术语 glossary

|  |  |
| --- | --- |
| 术语 | 简单解释 |
| Anchor peer |  |
| ACL |  |
| Block |  |
| Chain |  |
| Chaincode |  |
| Channel |  |
| Commitment |  |
| Concurrency Control Version Check |  |
| Configuration Block |  |
| Consensus |  |
| Consortium |  |
| Current State |  |
| Dynamic Membership |  |
| Endorsement |  |
| Endorsement Policy |  |
| Hyperledger Fabric CA |  |
| Genesis Block |  |
| Gossip Protocol |  |
| Initialize |  |
| Install |  |
| Instantiate |  |
| Invoke |  |
| Leading Peer |  |
| Ledger |  |
| Member |  |
| Membership Service Provider |  |
| Membership services |  |
| Ordering Service |  |
| Organization |  |
| Peer |  |
| Policy |  |
| Private Data |  |
| Private Data Collection |  |
| Proposal |  |
| Query |  |
| Software Development Kit(SDK) |  |
| Smart Contract |  |
| State Database |  |
| System Chain |  |
| Transaction |  |
| World State |  |

Terminology is important, so that all Hyperledger Fabric users and developers agree on what we mean by each specific term.

术语非常重要，所有Fabric用户和开发认可每个特定术语的描述。

What is a smart contract for example. The documentation will reference the glossary as needed, but feel free to read the entire thing in one sitting if you like;

比如什么叫智能合约,文档根据需要引用术语表，如果您喜欢可以一次性阅读完毕。

it’s pretty enlightening!

这是相当于有启发

##### Anchor peer

Used to initiate gossip communication between peers from different organizations.

为了不同组织的节点间建立通信。

The anchor peer serves as the entry point for another organization’s peer on the same channel to communicate with each of the peers in the anchor peer’s organization.

锚定节点是作为相同子链不同组织节点间通信的入口服务。

Cross-organization gossip is scoped to channels.

跨组织通信智能在子链之间。

In order for cross-org gossip to work, peers from one organization need to know the address of at least one peer from another organization in the channel.

为了实现跨组织通信，每个组织节点至少配置一个peer节点用于通信。

Each organization added to a channel should identify at least one of its peers as an anchor peer (there can be more than one).

每个组织加入一个子链，至少为其添加一个锚定节点便于不同节点间通讯身份校验(可以配置多个)。

The anchor peer address is stored in the configuration block of the channel.

锚定节点地址存储在子链的配置中。

##### ACL

An ACL, or Access Control List, associates access to specific peer resources (such as system chaincode APIs or event services) to a Policy (which specifies how many and what types of organizations or roles are required).

ACL(访问控制列表)，将对等节点资源(比如链码api和事件服务)和关联策略(哪些特定资源和角色允许)起来。

The ACL is part of a channel’s configuration.

ACL 是子链配置的一部分。

It is therefore persisted in the channel’s configuration blocks, and can be updated using the standard configuration update mechanism.

这些被持久化在链码配置块中，可以通过标准的更新机制进行更新。

An ACL is formatted as a list of key-value pairs, where the key identifies the resource whose access we wish to control, and the value identifies the channel policy (group) that is allowed to access it.

ACL被格式化为一系列键值对，那些键资源是系统控制资源，那些值标识控制允许访问策略。

For example lscc/GetDeploymentSpec: /Channel/Application/Readers defines that the access to the life cycle chaincode GetDeploymentSpec API (the resource) is accessible by identities which satisfy the /Channel/Application/Readers policy.

例如 lscc/GetDeploymentSpec: /Channel/Application/Readers 定义生命周期的链码开发的通过./Channel/Application/Readers安全策略的API

A set of default ACLs is provided in the configtx.yaml file which is used by configtxgen to build channel configurations.

configtx.yaml定义了一组默认的ACLs，configtx.xml使用yaml文件来构建通道配置。

The defaults can be set in the top level “Application” section of configtx.yaml or overridden on a per profile basis in the “Profiles” section.

默认值可以在“Application”部分顶级设置，也可以在每个配置中“Profiles”中重写。

##### Block

An ordered set of transactions that is cryptographically linked to the preceding block(s) on a channel.

在一个子链中，一些有序事务通过加密方式连接到前一个快。

##### Chain

The ledger’s chain is a transaction log structured as hash-linked blocks of transactions.

分布式链是个的事务日志，该结构以hash链接区块链事务。

Peers receive blocks of transactions from the ordering service, mark the block’s transactions as valid or invalid based on endorsement policies and concurrency violations, and append the block to the hash chain on the peer’s file system.

Peer端从排序服务接收事务区块，根据背书策略或当前验证结果，决定区块是否有效，并将区块HASH链接到peer文件系统中。

##### Chaincode

See Smart-Contract. 见 See Smart-Contract.

##### Channel

A channel is a private blockchain overlay which allows for data isolation and confidentiality.

子链是一个对区块数据隔离性和机密性的私有覆盖层

A channel-specific ledger is shared across the peers in the channel, and transacting parties must be properly authenticated to a channel in order to interact with it.

特定子链账本在子链中不同节点之间共享，为了与通道进行交流，相互之间必须进行身份验证。

Channels are defined by a Configuration-Block.

子链在区块配置中定义。

##### Commitment

Commitment 承诺 保障

Each Peer on a channel validates ordered blocks of transactions and then commits (writes/appends) the blocks to its replica of the channel Ledger.

通道上的每个对等点验证有序的事务块，然后将这些块提交(写/追加)到链账本的副本上。

Peers also mark each transaction in each block as valid or invalid.

Peer节点同样标记区块链中事务有效或无效。

##### Concurrency Control Version Check

##### **Configuration Block**

##### Consensus

##### Consortium

##### Current State

See World-State. 见World-State.

##### Dynamic Membership

##### Endorsement

##### Endorsement Policy

##### Hyperledger Fabric CA

##### Genesis Block

##### Gossip Protocol

##### Initialize

##### Install

##### Instantiate

##### Invoke

##### Leading Peer

##### Ledger

##### Member

##### Membership Service Provider

##### Membership services

##### Ordering Service

##### Organization

##### **Peer**

##### Policy

##### Private Data

##### Private Data Collection

##### Proposal

##### Query

##### Software Development Kit(SDK)

##### Smart Contract

##### State Database

##### System Chain

##### Transaction

##### World State

## 11.2 提问 still have questions

## 11.3 各版本简介

### 11.3.1 1.0 版本介绍

### 11.3.2 1.1 版本介绍

### 11.3.3 1.2 版本介绍

New major features

主要新特征

1 Private Data Collections: 私人数据收集

A way to keep certain data/transactions confidential among a subset of channel members. We also have an architecture document on this topic which can be found here.

channel成员子集之间保持一种可信的数据/事务通信。

2 Service Discovery: 服务发现

Discover network services dynamically, including orderers, peers, chaincode, and endorsement policies, to simplify client applications.

动态的网络服务发现，包括排序节点、peer节点、合约、背书策略、以及简单客户端应用。

3 Access control: 访问控制

How to configure which client identities can interact with peer functions on a per channel basis.

通过配置可以达到客户端身份与每个通道上的对等函数交互。

4 Pluggable endorsement and validation:可插拔的背书和验证

Utilize pluggable endorsement and validation logic per chaincode.

对每个链代码使用可插入背书和验证逻辑

New tutorials 新教程

1 Upgrade to version v1.2:升级到v1.2

Leverages the BYFN network to show how an upgrade flow should work. Includes both a script (which can serve as a template for upgrades), as well as the individual commands.

利用BYFN网络展示了一个升级流程，包括了一个脚本（作为升级模板），作为单个命令。

2 CouchDB: CouchDB

How to set up a CouchDB data store (which allows for rich queries).

如何设置CouchDB作为存储（丰富的查询支持）。

3 Private data: 私有数据

Shows how to set up a collection using BYFN.

演示了如何使用BYFN设置集合

4 Query certificates based on various filter criteria (Fabric CA): 基于过滤标准(Fabric CA)的查询证书

Describes how to use fabric-ca-client to manage certificates.

描述了如何使用fabric-ca-client去管理证书

Other new documentation 另外新文档

Service Discovery CLI:服务发现CLI

Configuring the discovery service using the CLI.

使用CLI配置服务发现

Release notes 发布说明

For more information, including FAB numbers for the issues and code reviews that made up these changes (in addition to other hygiene/performance/bug fixes we did not explicitly document), check out the:

更多信息,包括问题的FAB编号和构成这些更改的代码审查,请参与如下地址

1 Fabric release notes.

https://github.com/hyperledger/fabric/releases/tag/v1.2.0

2 Fabric CA release notes.

https://github.com/hyperledger/fabric-ca/releases/tag/v1.2.0

### 11.3.4 1.3 版本介绍

## 11.4 Fabric CA

见 fabric ca文档

# 词汇

consensus protocol 一致性协议

Cryptocurrency 加密货币

Permission 许可

underlying technology 底层技术

innovative enterprise 创新型企业

performance characteristics 性能特征

hard requirement 硬需求

Anti-Money Laundering (AML) 反洗钱

Know-Your-Customer (KYC) 了解客户

motivation  动机

distributed ledger technology (DLT)  分布式账本技术

domain-specific languages (DSL) 领域特定语言

anonymous  匿名

[crash fault-tolerant](https://en.wikipedia.org/wiki/Fault_tolerance) (CFT) 故障容错

byzantine fault tolerant (BFT) 拜占庭容错

“proof of work” (PoW). 工作量证明

malicious 恶意的

resiliency 弹性

resilienpotentially malicious contracts 潜在恶意代码

non-determinism 不确定性

The benchmarking efforts 基准测试

Zero knowledge proofs (ZKP) 零知识证明