浅谈Hyperledger Fabric

Hyperledger Fabric介绍

- Hyperledger是Linux基金会于2015年成立的跨行业的区块链解决方案。
- Fabric是其子项目之一，目前最成熟，其他的子项目包括：Sawtooth、Burrow、Composer等。
- Fabric区别于其他区块链的特点：private和permissioned，往往用于建立联盟链。

以太坊和Fabric的比较

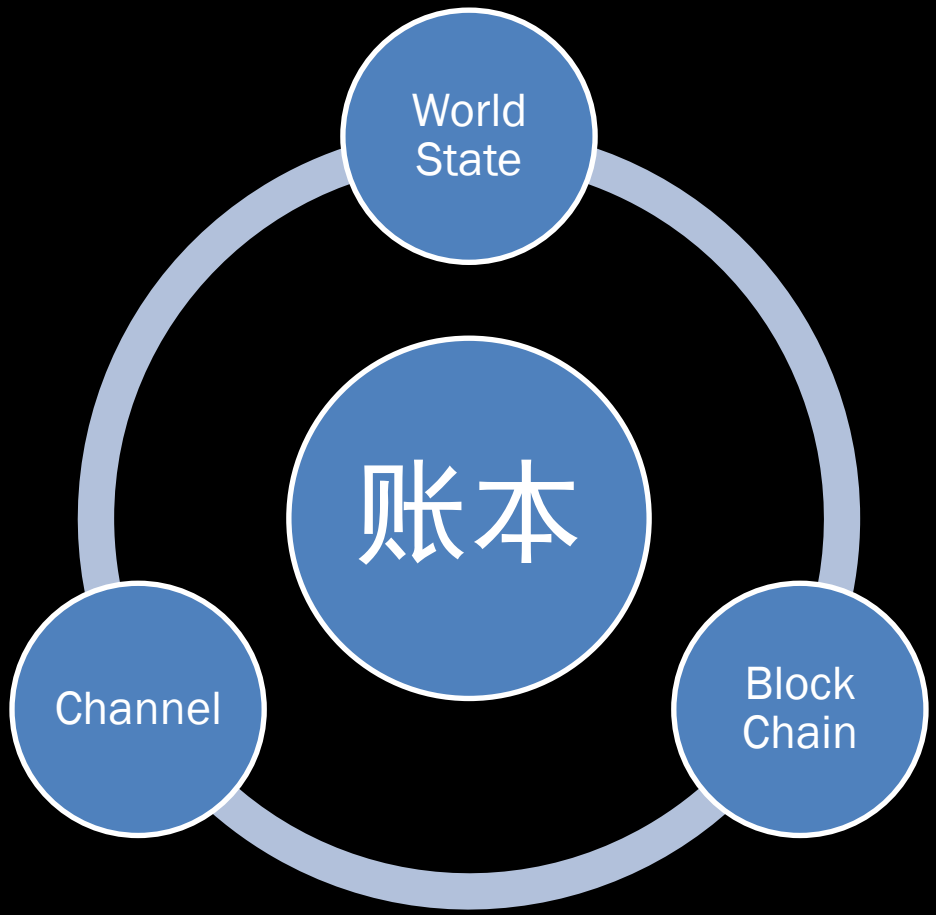
特性	以太坊	Fabric
平台描述	通用的区块链平台	模块化的区块链平台
管理方	以太坊开发者	Linux基金会
运营方式	无授权、公有、私有	有授权、私有
共识机制	• 基于POW的挖矿 • 账本层面	• 支持多种方式、更广泛的共识理解 • 交易层面
智能合约	Solidity	Go、Node
货币	• 以太 • 通过智能合约的代币	• 无 • 通过链码（Chaincode）的代币

参考：https://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda-21c1bb9442f6

Hyperledger Fabric组件分类

- 分布式账本，解决数据共享问题，让所有参与方拥有共同的交易历史。
- 智能合约，解决应用与账本的交互问题，包括查询和更新。
- 共识机制，解决数据同步问题，基于Endorsement Policy实现交易的传播。
- 成员服务，解决参与方身份问题，只有可信成员之间才能完成交易。

分布式账本



智能合约

- Chaincode，使用通用编程语言（Go、Node、Java）编写
  - Normal Chaincode
  - System Chaincode
- Chaincode != Fabric SDK
  - Chaincode运行于Fabric环境（Peer节点），负责账本更新
  - Client app通过Fabric SDK与之交互
- Chaincode的生命周期：package、install、instantiate、upgrade

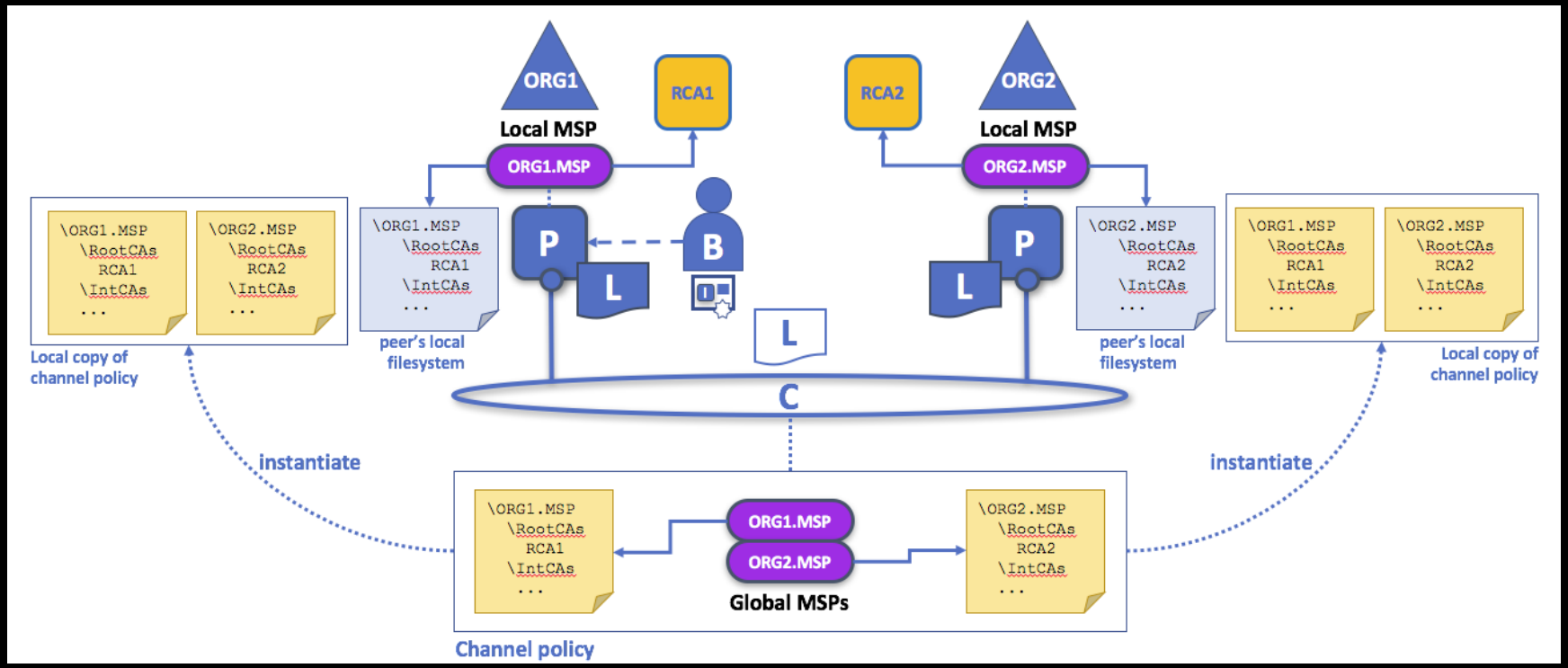
共识机制

- 涉及三方协调
  - Client、Peer、Orderer
- 三阶段协议
  - Proposal/Endorsement，Client < -- > Endorser，chaincode在此阶段执行
  - Ordering，Client < -- > Orderer
  - Validation/Commit，Orderer < -- > Peer

成员服务（MPS）

- Fabric中参与方的身份基于基于数字证书和PKI层次模型。
  - 即：决定能否进入Fabric
- MPS解决了参与方的受信问题，即哪些证书允许参与到当前交易（基于Channel）中来。
  - 即：决定能否参与到当前交易（Channel）
- MPS的层级
  - Local MPS（每个Peer）
  - Channel MPS

成员服务（MPS）的层级


ORG1
Local MSP
ORG1.MSP
RCA1
RCA2
ORG2
Local MSP
ORG2.MSP
\ORG1.MSP
\RootCAs
RCA1
\IntCAs
...
\ORG2.MSP
\RootCAs
RCA2
\IntCAs
...
P
L
B
C
peer's local filesystem
Local copy of channel policy
instantiate
ORG1.MSP
ORG2.MSP
Global MSPs
Channel policy

Fabric项目组成

- 业务逻辑（运行于Fabric环境）
  - Asset（Model）
  - Channel（数据隔离）
  - MPS（权限）
  - Chaincode（Business Service）
- Client App

使用Composer加速Fabric应用开发

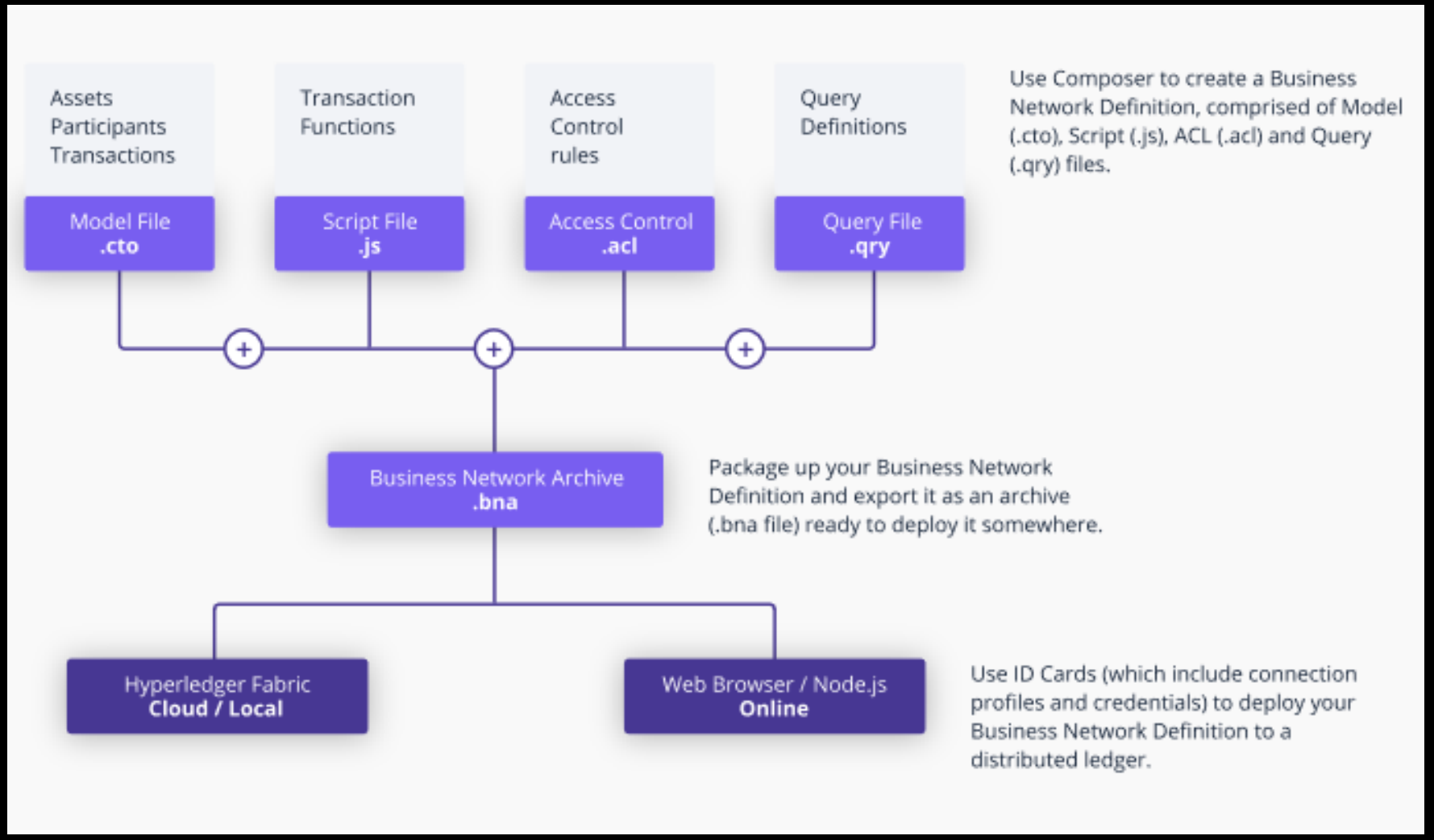
Hyperledger项目现在已经发展成包含一系列子项目的大工程。针对不同业务需求，可以支持不同的区块链实现方式。例如，Burrow专门用来实现带权限控制的Ethereum，而Indy更专注于数字身份。在这些子项目中，Fabric是最成熟的一个。当开发者们谈到使用 Hyperledger 技术时，实际上大多数时候是在考虑 Hyperledger Fabric。然而，chaincode的编程抽象相对底层，因为它直接处理账本的状态数据。此外，在编写第一行区块链代码之前，搭建基础设施也经常耗去很多时间。**HYPERLEDGER COMPOSER** 构建于Fabric基础之上，加速了将想法实现为软件的过程。Composer 提供 DSLs 来建立业务资源模型、定义访问控制和构建业务网络。使用 Composer，可以在不搭建任何基础设施的情况下，仅通过浏览器来验证我们的想法。需要明确的是，Composer 本身并不是区块链，仍然需要把它部署在 Fabric 上。

最新一期TW雷达对Composer的评语

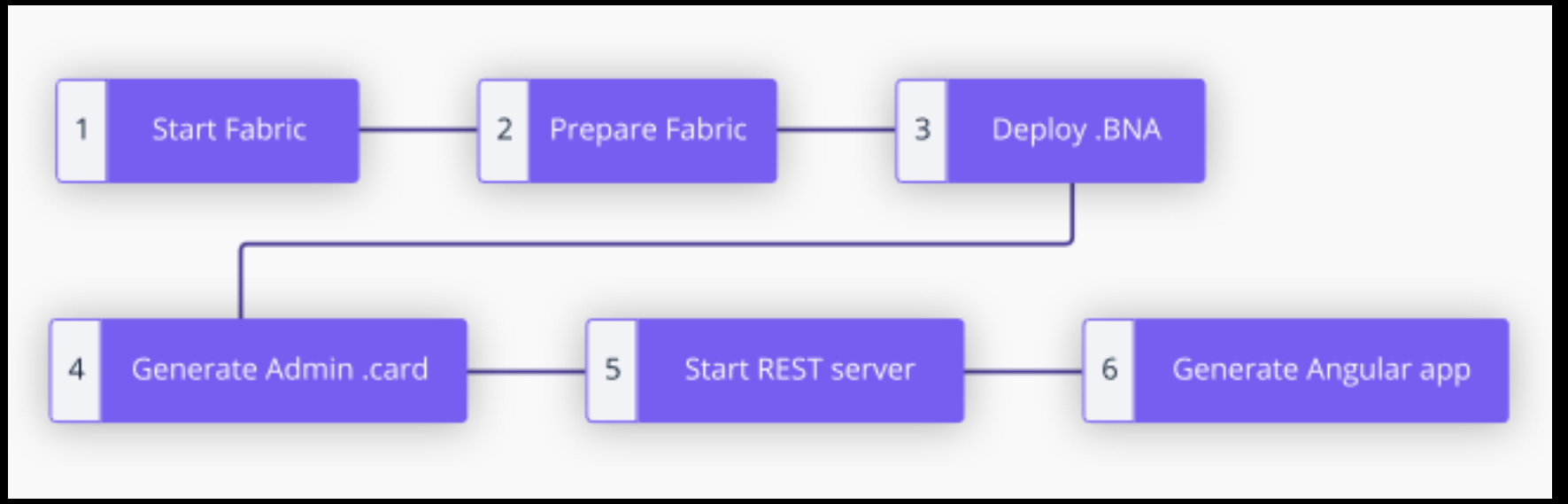
Composer的优势

- 文档条理性高
- 开发者友好
  - 建议按照Composer文档安装Fabric
  - 良好的抽象，避免直接书写Chaincode
  - 统一的工程化体验
  - 内建测试支持
  - CLI统一了开发、管理和运维任务
- 比原生Chaincode减少10+代码

业务逻辑的开发


Assets
Participants
Transactions
Transaction
Functions
Access
Control
rules
Query
Definitions
Use Composer to create a Business Network Definition, comprised of Model (.cto), Script (.js), ACL (.acl) and Query (.qry) files.
Model File
.cto
Script File
.js
Access Control
.acl
Query File
.qry
Business Network Archive
.bna
Package up your Business Network Definition and export it as an archive (.bna file) ready to deploy it somewhere.
Hyperledger Fabric
Cloud / Local
Web Browser / Node.js
Online
Use ID Cards (which include connection profiles and credentials) to deploy your Business Network Definition to a distributed ledger.

客户端的开发


1
Start Fabric
2
Prepare Fabric
3
Deploy .BNA
4
Generate Admin .card
5
Start REST server
6
Generate Angular app

业务逻辑：Model

```
/**
 * My commodity trading network
 */
namespace org.example.mynetwork
asset Commodity identified by tradingSymbol {
    o String tradingSymbol
    o String description
    o String mainExchange
    o Double quantity
    --> Trader owner
}
participant Trader identified by tradeId {
    o String tradeId
    o String firstName
    o String lastName
}
transaction Trade {
    --> Commodity commodity
    --> Trader newOwner
}
```

安装： https://hyperledger.github.io/composer/latest/installing/installing-index
教程： https://hyperledger.github.io/composer/latest/tutorials/tutorials

业务逻辑：ACL文件

```
/**
 * Access control rules for tutorial-network
 */
rule Default {
    description: "Allow all participants access to all resources"
    participant: "ANY"
    operation: ALL
    resource: "org.example.mynetwork.*"
    action: ALLOW
}

rule SystemACL {
  description:  "System ACL to permit all access"
  participant: "ANY"
  operation: ALL
  resource: "org.hyperledger.composer.system.**"
  action: ALLOW
}
```

安装： https://hyperledger.github.io/composer/latest/installing/installing-index
教程： https://hyperledger.github.io/composer/latest/tutorials/tutorials

业务逻辑：Query

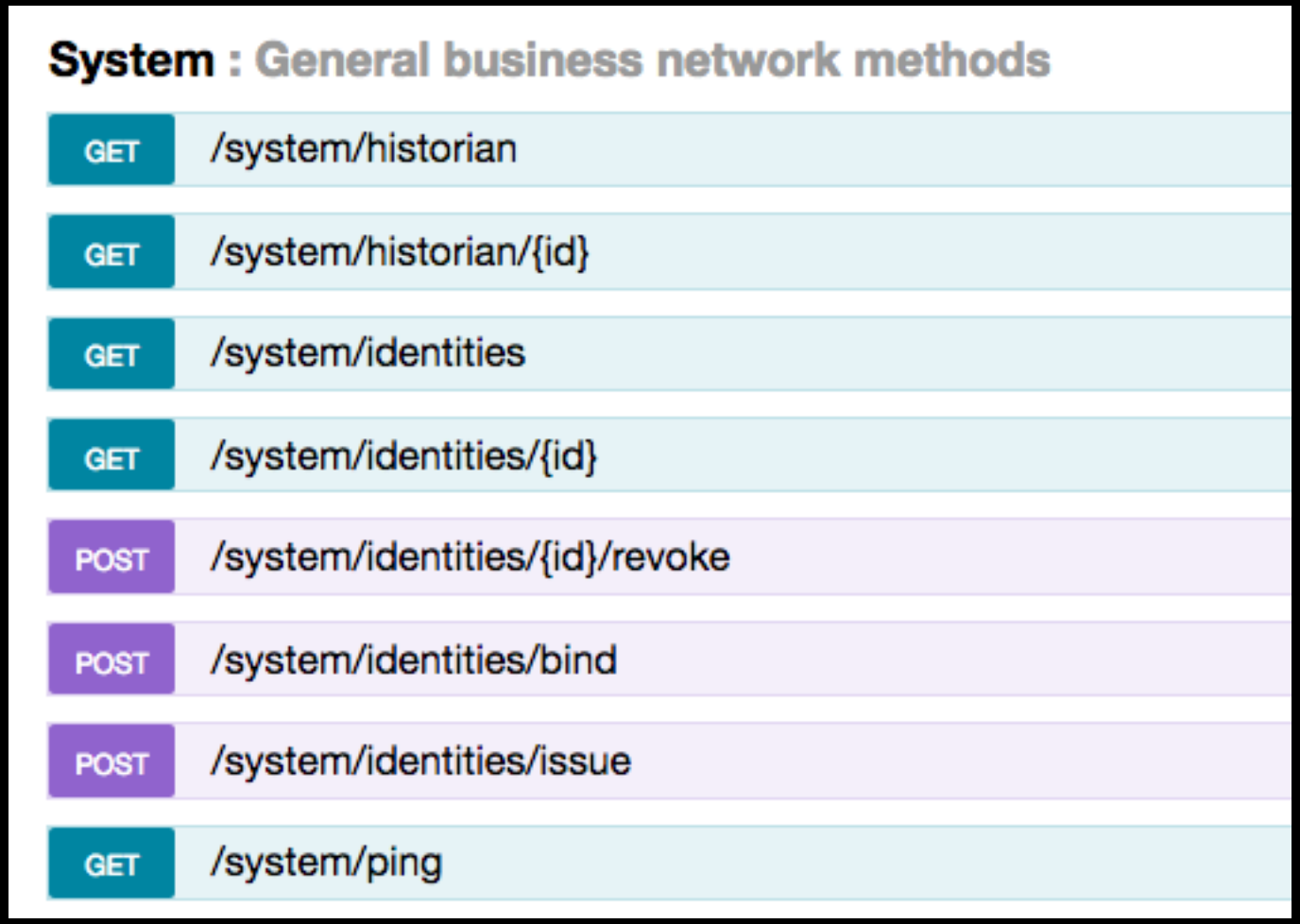
```
query selectCommodities {
  description: "Select all commodities"
  statement:
      SELECT org.example.mynetwork.Commodity
}

query selectCommoditiesByExchange {
  description: "Select all commodities based on their main exchange"
  statement:
      SELECT org.example.mynetwork.Commodity
          WHERE (mainExchange==_$exchange)
}

query selectCommoditiesByOwner {
  description: "Select all commodities based on their owner"
  statement:
      SELECT org.example.mynetwork.Commodity
          WHERE (owner == _$owner)
}
```

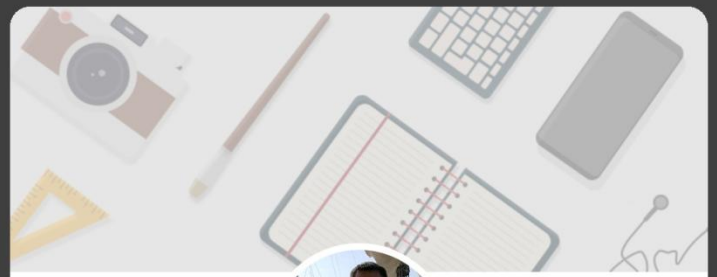
安装： https://hyperledger.github.io/composer/latest/installing/installing-index
教程： https://hyperledger.github.io/composer/latest/tutorials/tutorials

Rest Server



安装： https://hyperledger.github.io/composer/latest/installing/installing-index
教程： https://hyperledger.github.io/composer/latest/tutorials/tutorials

FAQ



胡键

写了48361字，获得了95个喜欢

前500强员工、CSM、咨询师，架构师、创业者、译者、社区组织者和开源软件深度参与者（Grails、Vert.x贡献者和dgate作者）。现专注于工业物联网创业，擅长围绕设备资产和生产管理提供物联网端到端解决方案。对海量设备接入、数据存储和数据分析有深刻理解。对于物联网、大数据、微服务、机器学习和MES有深入的研究。涉足领域包括：港口生产调度和EAM、煤炭生产和EAM、交通运政、工业物联网平台建设（含接入网关）。服务对象主要为国内相关行业领导者。

联系邮箱：jian.hu@shifudao.com