Hyperledger Fabric Fundamentals

Swetha Repakula
Open Source Contributor, IBM Open Technologies
srepaku@us.ibm.com
@swayr93

Morgan Bauer
Open Source Contributor, IBM Open Technologies
mbauer@us.ibm.com
@ibmhb
TradeLens improves **global trade efficiency**

- TradeLens is an open, extensible platform for sharing shipping events, messages, and documents across all the actors and systems in the supply chain ecosystem.

- It provides shared visibility and shared state for container shipments

**Benefits**

- Increase speed and transparency for cross border transactions through real time access to container events.

- Reduced cost and increased efficiency through paperless trade
Food Trust

What?

• Provide a trusted source of information and traceability to improve transparency and efficiency across the food network.

How?

• Shared ledger for storing digital compliance documentation, test results and audit certificates network.

Benefits

• Reduce impact of food recalls through instant access to end-to-end traceability data to verify history in the food network and supply chain.
• Help to address the 1 in 10 people sickened and 400,000 fatalities which occur every year from food-born illnesses.
Further examples by (selected) industry

<table>
<thead>
<tr>
<th>Financial</th>
<th>Public Sector</th>
<th>Retail</th>
<th>Insurance</th>
<th>Manufacturing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Trade Finance</td>
<td>• Asset Registration</td>
<td>• Supply chain</td>
<td>• Claims processing</td>
<td>• Supply chain</td>
</tr>
<tr>
<td>• Cross currency payments</td>
<td>• Citizen Identity</td>
<td>• Loyalty programs</td>
<td>• Risk provenance</td>
<td>• Product parts</td>
</tr>
<tr>
<td>• Mortgages</td>
<td>• Medical records</td>
<td>• Information sharing (supplier – retailer)</td>
<td>• Asset usage history</td>
<td>• Maintenance tracking</td>
</tr>
<tr>
<td>• Letters of Credit</td>
<td>• Medicine supply chain</td>
<td></td>
<td>• Claims file</td>
<td></td>
</tr>
</tbody>
</table>
For the last 17 years, The Linux Foundation® has provided unparalleled support for open source communities through financial and intellectual resources, governance structure, IT infrastructure, services, events, and training.

Dedicated to building sustainable ecosystems around open source projects, The Linux Foundation is working with the global technology community to solve the world’s hardest problems through open source and creating the largest shared technology investment in history.

The Linux Foundation is the umbrella organization for more than 60 open source projects accelerating open technology development and commercial adoption. Some of the game-changing initiatives hosted by The Linux Foundation include:
Hyperledger Modular Approach

Distributed Ledgers
- Hyperledger Burrow: Permissionable smart contract machine (EVM)
- Hyperledger Fabric: Permissioned with channel support
- Hyperledger Indy: Decentralized identity
- Hyperledger Iroha: Mobile application focus
- Hyperledger Sawtooth: Permissioned & permissionless support; EVM transaction family

Libraries
- Hyperledger Aries
- Hyperledger Quilt
- Hyperledger Transact
- Hyperledger Ursa

Tools
- Hyperledger Caliper
- Hyperledger Cello
- Hyperledger Composer
- Hyperledger Explorer

Domain-Specific
- Hyperledger Grid
- Hyperledger Labs
What is Hyperledger Fabric

• An implementation of blockchain technology that is intended as a foundation for developing blockchain applications for the enterprise

• Key characteristics:
  – Permissioned
  – Highly modular
    • Pluggable consensus, ledger, membership services, endorsement and validation
  – Smart contracts in general purpose languages
  – Privacy
  – No “mining” or native crypto-currency required for consensus
  – Execute-order-validate vs order-execute
Technical Deep Dive

- [Architectural Overview]
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap
Hyperledger Fabric V1 Architecture

- Client Application
- SDK (HFC)
- Membership Services
- Fabric-CA
- External-CA
- Peer
  - Endorser
  - Committer
    - Ledger
    - Chaincode
    - Events
  - Ordering-Service
  - Hyperledger Fabric Network

IBM Blockchain
How applications interact with the ledger

1. Client Application in using Hyperledger Fabric Client (HFC) SDK
2. Smart Contract implemented using chaincode – managing the World state
The Fabric ledger is maintained by each peer and includes the blockchain and worldstate. A separate ledger is maintained for each channel the peer joins. Transaction read/write sets are written to the blockchain. Channel configurations are also written to the blockchain. The worldstate can be either LevelDB (default) or CouchDB. LevelDB is a simple key/value store, and CouchDB is a document store that allows complex queries. The smart contract Contract decides what is written to the worldstate.
Technical Deep Dive

- Architectural Overview
- [ Network Consensus ]
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap
## Nodes and roles

<table>
<thead>
<tr>
<th>Node Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Committing Peer</strong></td>
<td>Maintains ledger and state. Commits transactions. May hold smart contract.</td>
</tr>
<tr>
<td><strong>Endorsing Peer</strong></td>
<td>Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.</td>
</tr>
<tr>
<td><strong>Ordering Node</strong></td>
<td>Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.</td>
</tr>
</tbody>
</table>
Hyperledger Fabric Consensus

Consensus is achieved using the following transaction flow:

Endorse → Order → Validate
Sample transaction: Step 1/7 – Propose transaction

Application proposes transaction

Endorsement policy:
- "E₀, E₁, and E₂ must sign"
- (P₃, P₄ are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers \{E₀, E₁, E₂\}

Key:
- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Ordering Service
- Smart Contract (Chaincode)
- Endorsement Policy
Sample transaction: Step 2/7 – Execute proposal

Endorsers Execute Proposals

E₀, E₁ & E₂ will each execute the proposed transaction. None of these executions will update the ledger.

Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:

- **Endorser**
- **Ledger**
- **Comitting Peer**
- **Application**
- **Ordering Node**
- **Endorsement Policy**
- **Smart Contract (Chaincode)**
Sample transaction: Step 3/7 – Proposal Response

Application receives responses

RW sets are asynchronously returned to application

The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Ordering Service**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Sample transaction: Step 4/7 – Order Transaction

Responses submitted for ordering

Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications.

Key:
- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Smart Contract (Chaincode)
- Endorsement Policy

Hyperledger Fabric Network

(Other applications)
Sample transaction: Step 5/7 – Deliver Transaction

Ordering service collects transactions into proposed blocks for distribution to committing peers. Peers can deliver to other peers in a hierarchy (not shown).

Different ordering algorithms available:
- SOLO (Single node, development)
- Kafka (Crash fault tolerant)
- Raft (Crash fault tolerant)

Key:
- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Chaincode
- Endorsement Policy
Sample transaction: Step 6/7 – Validate Transaction

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state.

Validated transactions are applied to the world state and retained on the ledger.

Invalid transactions are also retained on the ledger but do not update world state.

Key:
- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Ordering-Service
- Smart Contract (Chaincode)
- Endorsement Policy
Sample transaction: Step 7/7 – Notify Transaction

Committing peers notify applications

Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

Applications will be notified by each peer to which they are connected

Key:
- Endorser
- Ledger
- Committing Peer
- Application
- Ordering Node
- Ordering-Service
- Smart Contract (Chaincode)
- Endorsement Policy

Hyperledger Fabric Network
Technical Deep Dive

- Architectural Overview
- Network Consensus
- [Channels and Ordering Service]
- Components
- Network setup
- Endorsement Policies
- Membership Services
- Roadmap
Ordering Service

The ordering service packages transactions into blocks to be delivered to peers. Communication with the service is via channels.

Different configuration options for the ordering service include:

– **SOLO**
  - Single node for development

– **Kafka** : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended

– **Raft** : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended
Channels

Channels provide privacy between different ledgers

- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is **installed** on peers to access the worldstate
- Chaincode is **instantiated** on specific channels
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability
Single Channel Network

- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger.
- Endorsement by peers $E_0$, $E_1$, $E_2$ and $E_3$. 

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Multi Channel Network

- Peers $E_0$ and $E_3$ connect to the red channel for chaincodes $Y$ and $Z$
- $E_1$, $E_2$ and $E_3$ connect to the blue channel for chaincodes $A$ and $B$

Key:

- **Endorser**
- **Ledger**
- **Committing Peer**
- **Application**
- **Ordering Node**
- **Smart Contract (Chaincode)**
- **Endorsement Policy**
Technical Deep Dive

• Architectural Overview
• Network Consensus
• Channels and Ordering Service
• Components
• Network setup
• [ Endorsement Policies ]
• Membership Services
• Roadmap
An endorsement policy describes the conditions by which a transaction can be endorsed. A transaction can only be considered valid if it has been endorsed according to its policy.

- Each chaincode is deployed with an Endorsement Policy
- ESCC (Endorsement System ChainCode) signs the proposal response on the endorsing peer
- VSCC (Validation System ChainCode) validates the endorsements
Endorsement Policy Syntax

Policy Syntax: \texttt{EXPR(E, E...)}

Where \texttt{EXPR} is either AND or OR and \texttt{E} is either a principal or nested \texttt{EXPR}

Principal Syntax: \texttt{MSP.ROLE}

Supported roles are: member and admin

Where \texttt{MSP} is the MSP ID, and \texttt{ROLE} is either “member” or “admin”

Instantiate the chaincode \texttt{mycc} on channel \texttt{mychannel} with the policy \texttt{AND('Org1MSP.member')}
Endorsement Policy Examples

Examples of policies:

• Request 1 signature from all three principals
  – \text{AND('Org1.member', 'Org2.member', 'Org3.member')}

• Request 1 signature from either one of the two principals
  – \text{OR('Org1.member', 'Org2.member')}

• Request either one signature from a member of the Org1 MSP or (1 signature from a member of the Org2 MSP and 1 signature from a member of the Org3 MSP)
  – \text{OR('Org1.member', \text{AND('Org2.member', 'Org3.member')})}
Technical Deep Dive

- Architectural Overview
- Network Consensus
- Channels and Ordering Service
- Components
- Network setup
- Endorsement Policies
- Membership Services
- [ Roadmap ]
Based on [https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric+Roadmap](https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric+Roadmap) - Dates determined by the Hyperledger community - (*) Subject to change
Fabric 1.4+ planned features overview

• Higher level chaincode and client APIs
  – Brings to Fabric some of the Composer functionality and simplicity

• Idemix Node.js and Go SDK
  – Complete support for Identity Mixer in all SDKs

• Zero-Knowledge Asset Transfer (ZKAT) & Unspent Transaction Output (UTXO)
  – Privacy preserving asset/token transfer

• RAFT Consensus
  – Eliminates dependency on KAFKA and confirms pluggability of consensus

• Other possible future developments:
  – SBFT
Getting started with Hyperledger Fabric

- Build Your First Network (BYFN) – Network administrator
  - A simple network with 2 organizations running 2 peers, with one channel, a simple chaincode
  - Dockerhub images
  - Uses predefined enrollment certificates and «Solo» Ordering Service

- Extend Your First Network – Network administrator
  - Adds a 3rd organization to BYFN

- Develop Your First Application – Application developer
  - A simple Node.js application

- Start in devmode (minimal set up), then move to network (several peers), and security (membersrvc)

- Several examples to start from (fabcar)
Resources

  – Docs + tutorials

• IBM Code: https://developer.ibm.com/code/technologies/blockchain/
  – Code patterns, lectures, howtos, lab, etc

• IBM Blockchain Dev Center: https://developer.ibm.com/blockchain/
  – Blockchain 101

• IBM Blockchain Platform:
Thank you

Swetha Repakula
srepaku@us.ibm.com
@swayr93

Morgan Bauer
mbauer@us.ibm.com
@ibmhb

IBM Blockchain

www.ibm.com/blockchain
developer.ibm.com/blockchain
www.hyperledger.org

© Copyright IBM Corporation 2018.