IBM BLOCKCHAIN DEV DAY

Kangwuk Heo
Blockchain Leader, Client Innovation Lab,
CTO Office, IBM Korea
Sep 15, 2018

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

IBM

Hyperledger Fabric 1.1 and 1.2

Hyperledger Fabric 1.1

Hyperledger Fabric 1.1

- Node.js chaincode support- developers can now author chaincode using the most popular framework for the world's most popular programming language
- Channel based event service – to enable clients to subscribe to block and block transaction events on a per-channel basis.
- Ability to package CouchDB indexes with chaincode, to improve performance
- Ability to generate certificate revocation lists (CRLs)
- Ability to dynamically update client identities and affiliations
- Node.js SDK connection profiles to simplify connections to Fabric nodes
- Mutual Transport Layer Security (TLS) between Fabric nodes, and between clients and nodes
- Ability to encrypt ledger data for confidentiality using the chaincode encryption library
- Attribute-based Access Control in chaincode
- Chaincode APIs to retrieve client identity for access control decisions
- Performance improvements for transaction throughput and response time

Channel Events

Peers now deliver events per channel

Rearchitected in Fabric 1.1

Events captured during endorsement and stored in the ledger

Peers can replay past events

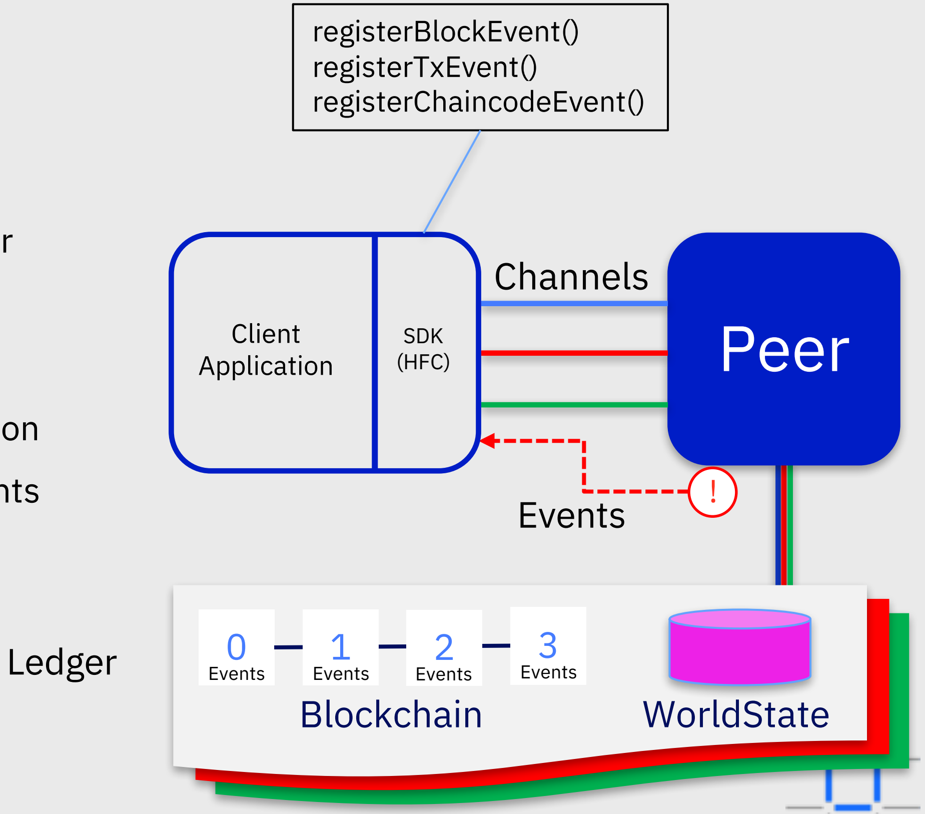
Applications can request old events to catch-up

Events can include the entire block or be filtered by transaction

Channel MSP defines which applications can register for events

Applications register listeners for:

- Block creation events
- Transaction events
- Custom chaincode events



Couch DB Indexes

Indexes can be packaged with chaincode to improve query performance

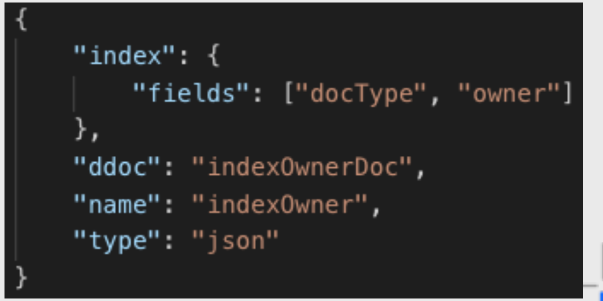
Indexes packaged alongside chaincode in the following directory:

– */META-INF/statedb/couchdb/indexes*

Each index must be defined separately in its own .json file

When chaincode is first installed, the index is deployed to CouchDB on instantiation

Once the index is deployed it will automatically be used by CouchDB



```
{
    "index": {
        "fields": ["docType", "owner"]
    },
    "ddoc": "indexOwnerDoc",
    "name": "indexOwner",
    "type": "json"
}
```

Chaincode

Chaincode contains business logic deployed to peers

Installed on peers and instantiated on channels

Run in secured docker images separate to the peer

Interact with the worldstate through the Fabric shim interface

Each chaincode has its own scoped worldstate

Language support for:

– Golang

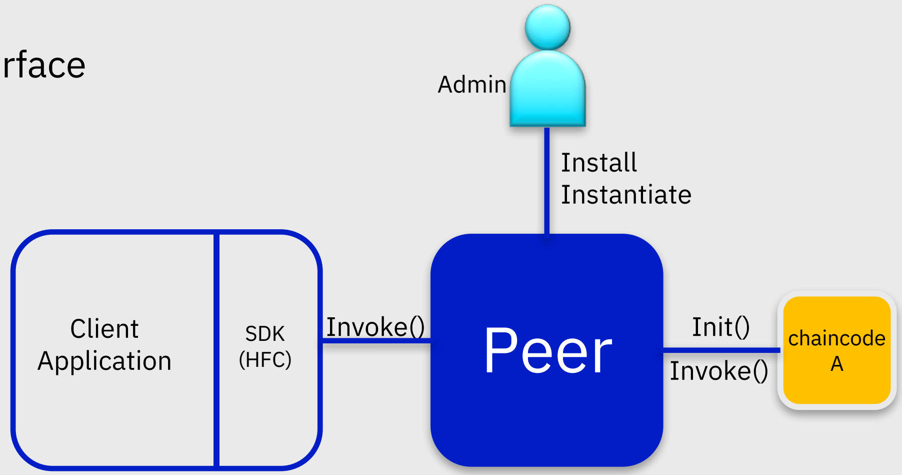
– Node.js (new in Fabric 1.1)

– Java (Future FAB-8063)

Implements:

– Init() - Called on instantiate and upgrade

– Invoke() – Called from client application



Transaport Layer Security

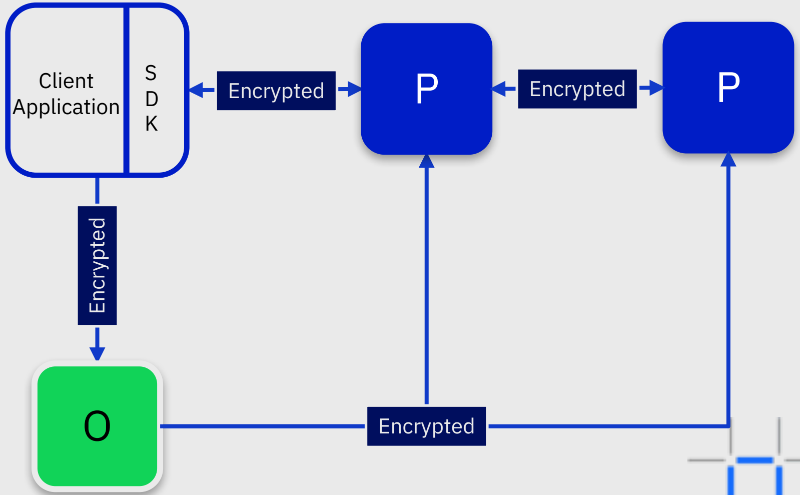
All communications within a Hyperledger Fabric network can be secured using TLS

Peers and Orderers are both TLS Servers and TLS Clients

Applications and commands are TLS Clients

Fabric 1.0.x support for TLS Server authentication

Fabric 1.1 supports mutual TLS (client authentication)



Hyperledger Fabric 1.2

Hyperledger Fabric 1.2

- Private Data Collections: A way to keep certain data/transactions confidential among a subset of channel members.
- Service Discovery: Discover network services dynamically, including orderers, peers, chaincode, and endorsement policies, to simplify client applications.
- Access control: How to configure which client identities can interact with peer functions on a per channel basis.
- Pluggable endorsement and validation: Utilize pluggable endorsement and validation logic per chaincode.

Private Data Collections

Allows data to be private to only a set of authorized peers

Fabric 1.0	Fabric 1.2
• Data privacy across channels only	• Data privacy within a channel
• Transaction proposal and worldstate read/write sets visible to all peers connected to a channel	• Transaction proposal and worldstate read/write sets available to only permissioned peers
• Ordering service has access to transactions including the read/write sets	• Ordering service has only evidence of transactions (hashes)
	• Complements Fabric 1.0 channel architecture
	• Policy defines which peers have private data

https://jira.hyperledger.org/browse/FAB-8718

Private Data Collections – Marble Scenario

Privacy Requirements:

- No marble data should go through ordering service as part of a transaction
- All peers have access to general marble information
  - *Name, Size, Color, Owner*
- Only a subset of peers have access to marble *pricing* information

Transaction
- Primary read/write set (if exists)
- Hashed private read/write set (hashed keys/values)

Transaction

Public channel data

Goes to all orderers/peers

Collection: Marbles
- Private Write Set
- Name, Size, Color, Owner

Policy: Org1, Org2
"requiredPeerCount": 1,
"maxPeerCount":2,
"blockToLive":1000000

Collection: Marbles
- Private data for channel peers
- Goes to all peers but not orderers

Collection: Marble Private Details
- Private Write Set
- Price

Policy: Org1
"requiredPeerCount": 1,
"maxPeerCount": 1,
"blockToLive":3

Collection: Marbles Private Details
- Private data for subset of channel peers
- Goes to subset of peers only

Hyperledger Fabric Roadmap

HYPERLEDGER FABRIC IBM

v1 GA	v1.1	v1.2	v1.3
▪ Docker images ▪ Tooling to bootstrap network ▪ Fabric CA or bring your own ▪ Channels for privacy ▪ Cross Channel Query ▪ Java and Node.js SDKs ▪ Ordering Services - Solo or Kafka ▪ Endorsement policy ▪ Level DB and Couch DB ▪ Block dissemination via Gossip ▪ Chaincode ACL ▪ Chaincode packaging & LCl ▪ HSM support ▪ Config transaction lifecycle ▪ Event security ▪ Peer management APIs	▪ Network administration: - Node.js connection profile ▪ Smart contract: - Node.js smart contracts - Encryption library - Attribute Based Access Control ▪ Performance & scale: - More orderers at scale - Parallel txn validation - CouchDB indexes ▪ Events: - Per channel vs global - Block info minimal events ▪ Membership services: - CSR for secure certificates ▪ Serviceability: - Upgrade from 1.0	▪ Network administration: - ACL mechanism per channel - Service discovery ▪ Consensus: - Pluggable endorsement and validation ▪ Smart Contract: - Private Data Collections (SideDB) ▪ Documentation: - Improved documentation and tutorials ▪ Serviceability: - Improvements and bug fixes	▪ Network administration: - CLI redesign - SDK improvements - Service Discovery remaining items ▪ Consensus: - State based endorsement ▪ Smart Contract: - Java chaincode - Burrow EVM support - Private Data remaining items - Chaincode query result pagination ▪ Membership services: - Identity Mixer ▪ Serviceability: - Improvements and bug fixes
July 2017	March 2018	June 2018	*Sept 2018* (quarterly)*

Based on https://wiki.hyperledger.org/projects/fabric/roadmap - Dates determined by the Hyperledger community - (*) Subject to change

Fabcoin

Fabcoin

Developed Aug 2017 in IBM Research – Zurich, originally for performane evaluation of Fabric

Fabcoin v1 (Oct 2017), 'single input, single output' coin

- Authority minted
- Implemented mostly as the custom Fabric VSCC (Validation System Chaincode)
- UTXO coin ownership model
- Used for performance evaluation
- Evaluated in Hyperledger Fabric paper submitted to Eurosys on an Oct 2017 Fabric master
  - 4500 tps MINT and 3000 tps SPEND performance on commodity hardware
  - Sub-second latencies/finality

Fabcoin v2 (Dec 2017), turing Fabcoin into a real token framework for Fabric

- Full-fledged 'multiple inut multiple output' (MIMO) coin
- Retained : UTXO coin ownership model, bulk of the logic coded in custom Fabric VSCC
- Callable/spendable from other chaincode
- Support for multiple 'central banks' (ie, minting authorities)
- Over 3500 tps for MINT/SPEND performance with Fabric v1.1-preview

Thank you

Kangwuk Heo
Blockchain Leader, Client Innovation Lab, CTO Office
—
kwheo@kr.ibm.com
+821049954958
ibm.com

개발자라면 지금 방문하세요!

developer.ibm.com/kr