Hyperledger Fabric

Introduction

Stefano Avola,

Email: s.avola@cipi.unige.it or s.avola@doc-space.net

Outline

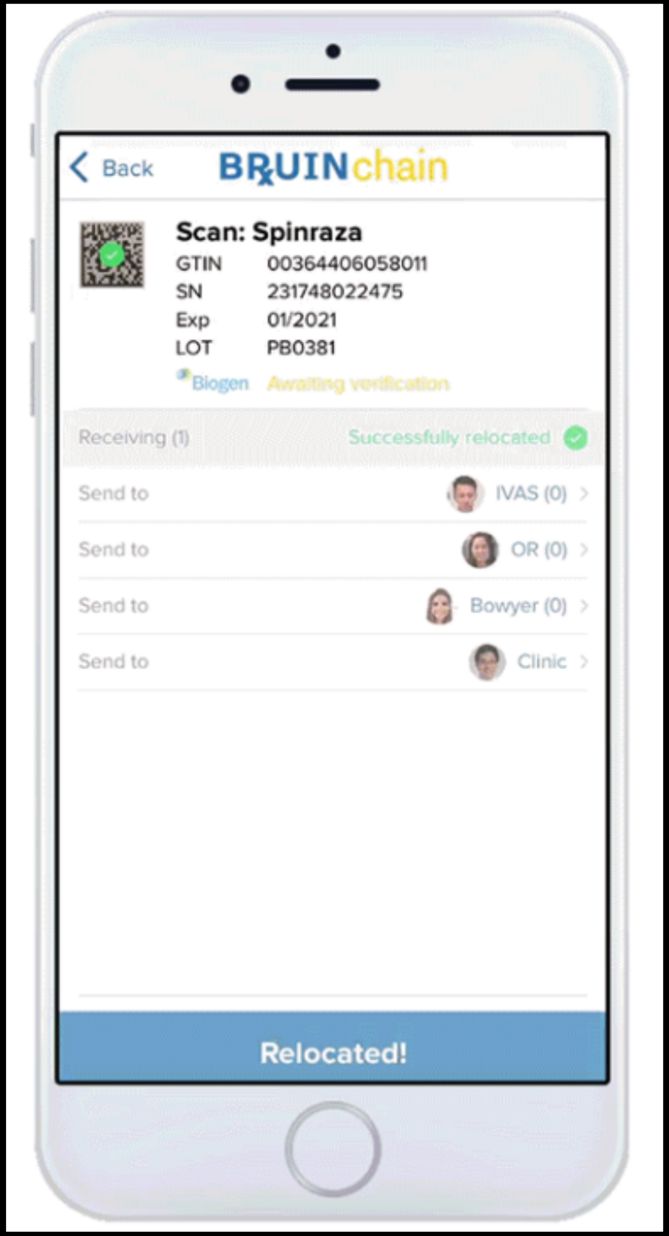
Introduction

- HLF is an open-source system for developing permissioned Blockchains. It is one of the Hyperledger projects conducted by the Linux Foundation.
  - The Hyperledger community borns at the end of 2015 thanks to about thirty vendor members of technology platforms and software houses, among which: Cisco, Fujitsu, Hitachi, IBM, Intel, NEC, NTT Data, Red Hat, VMware, SAP.

- "**Permissioned Blockchains**", that is "private" Blockchains where you have a specific identity and you need permissions (defined by the Blockchain owner) to be able to join them. Such owner could be, for example, a consortium, an association etc.
  - Unlike what happens in a permissionless system, where unknown identities can freely join the network, Hyperledger Fabric's members must enroll in the network via the so called Membership Service Provider (MSP)

Use cases

- Supply chain
  - Bruinchain
    - https://www.bruinchain.com/blockchain-would-save-pharmaceutical-industry-180-million-every-year
    - https://www.hyperledger.org/learn/publications/ledgerdomain-case-study
  - GSBN
    - https://www.hyperledger.org/case-studies/gsbn-case-study
- Banking
- Financial Services
- Healthcare
- IoT
- …
- https://www.hyperledger.org/learn/case-studies


Back BRUINchain

Scan: Spinraza

GTIN 00364406058011

SN 231748022475

Exp 01/2021

LOT PB0381

Biogen Awaiting verification

Receiving (1) Successfully relocated

Send to IVAS (0)

Send to OR (0)

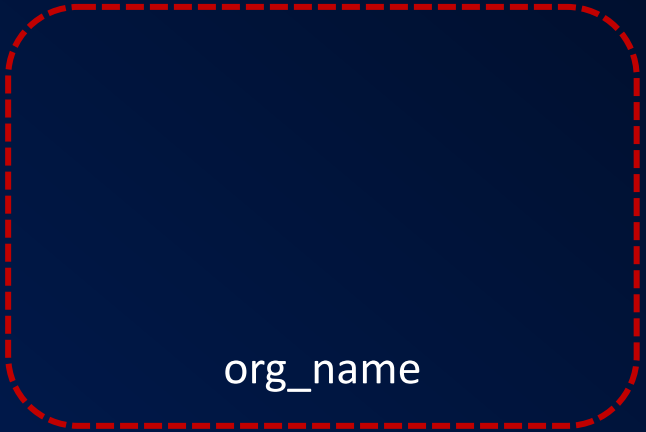
Send to Bowyer (0)

Send to Clinic

Relocated!

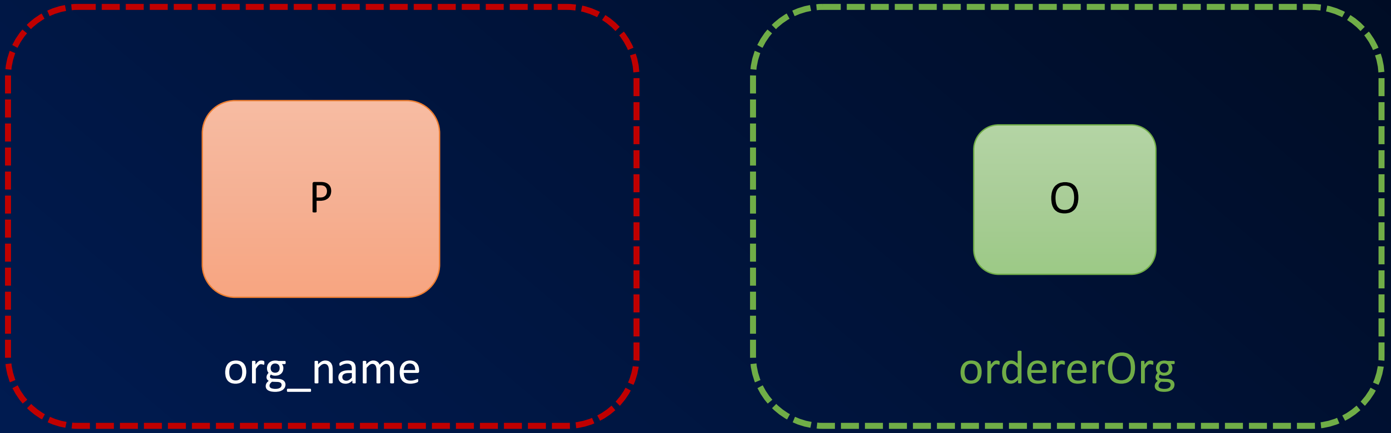
Organizations

- Also called «members»
- Are identities which can be as big as multinationals or as small as individuals
- Own peers and/or orderers



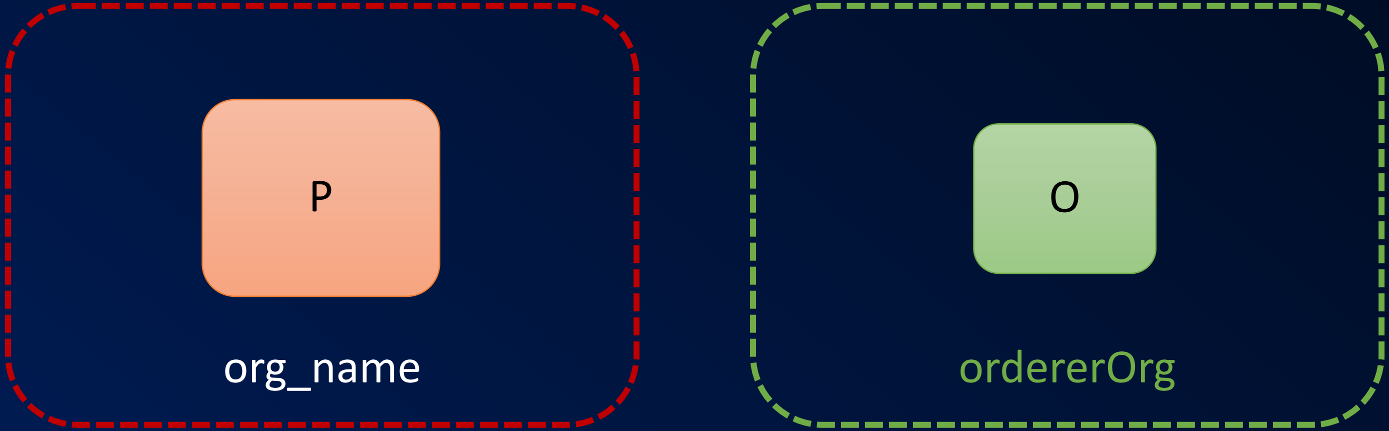
Peers and Orderers

- Blockchain network nodes
- Each Peer keeps a synchronized copy of the Ledger and interact with the client application to be able to execute transactions
- Leading, Endorsing, Anchor peer.



Peers and Orderers

- Each Orderer have the role of receiving transactions and creating blocks to disseminate to peers
- Leading orderer.



Channel

- Logical structure to allow peers and orderers to communicate each other
- It is related to 1 Ledger
- All the information about a channel are not visible nor accessible by nodes which are not connected to that channel

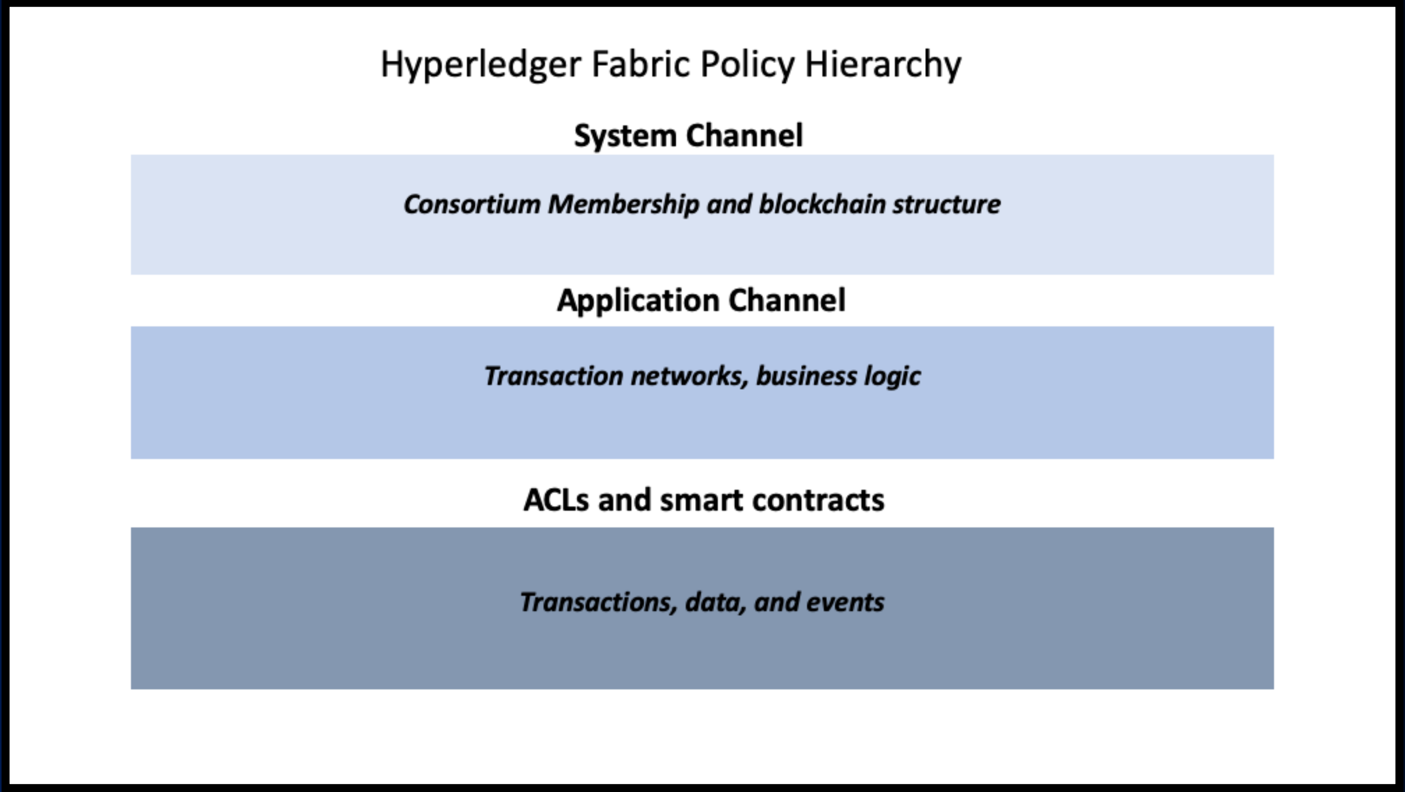
MSP

- Membership Service Provider is an abstract component providing credentials to clients, peers and orderers so that they can join the channel
- Provides a way to manage information about an identity, like public certificates or private keys
- Identities are similar to credit cards, which are used to show that you can pay. MSP is like a list of accepted credit cards

Identities
MSP
Accepted Here

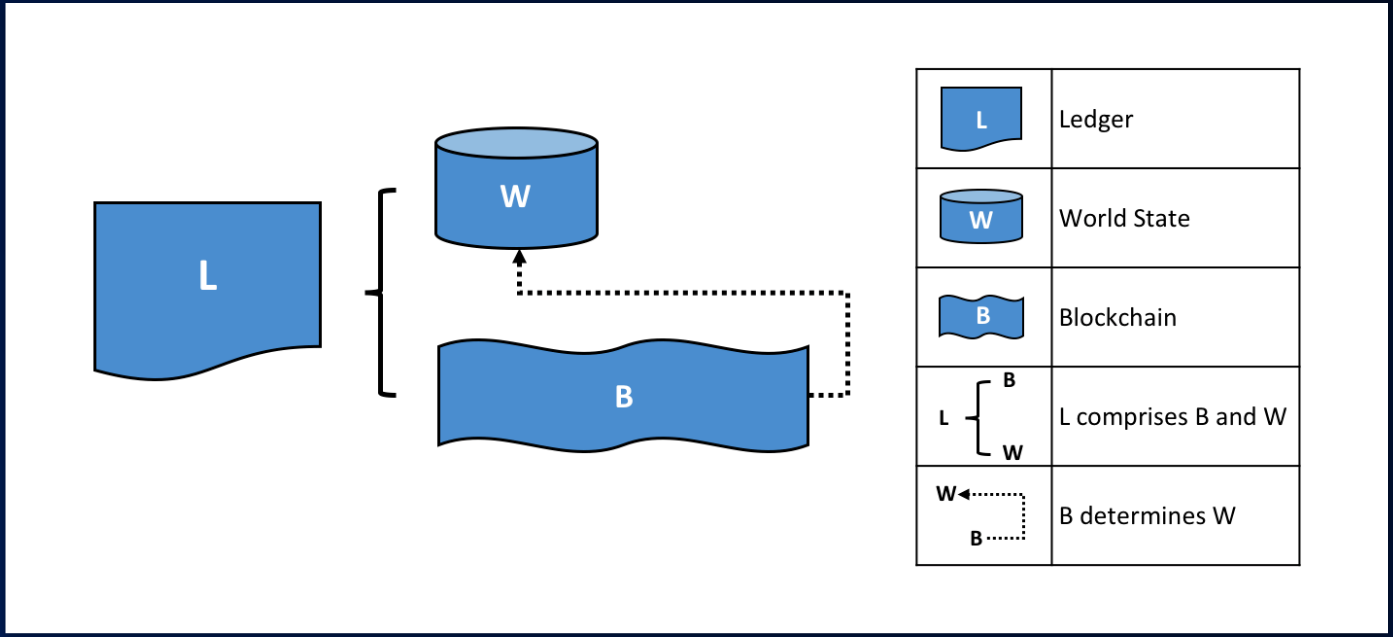
Policies

- Set of rules by which each operations in the network, especially its configuration ones, can be approved and then committed into the Blockchain.
- OR('Org1MSP.peer', 'Org2MSP.peer')


Hyperledger Fabric Policy Hierarchy
System Channel
Consortium Membership and blockchain structure
Application Channel
Transaction networks, business logic
ACLs and smart contracts
Transactions, data, and events

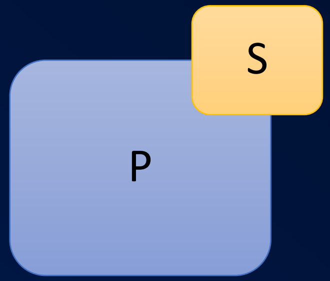
Ledger

- Contains data.
- Consists of:
  1. **Blockchain**, which saves all the data changes since the beginning of time.
  2. **World State**, which saves the actual state (value) of the data. It is a database: LevelDB or CouchDB (NoSQL).



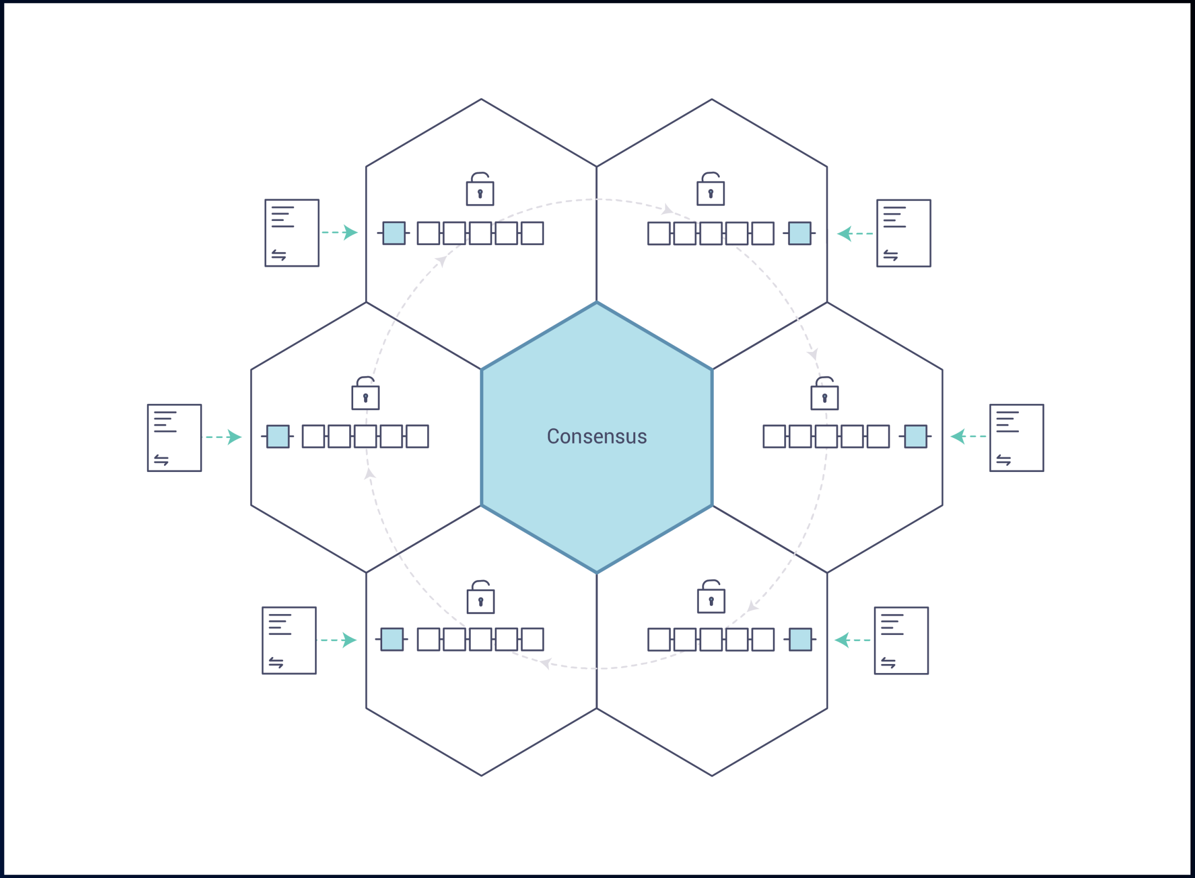
Smart Contract and Chaincode

- Smart Contracts are classes defining the business logic or the services with which the client application can interact.
- One Chaincode
  - is a set of Smart Contracts.
  - Is installed on peers



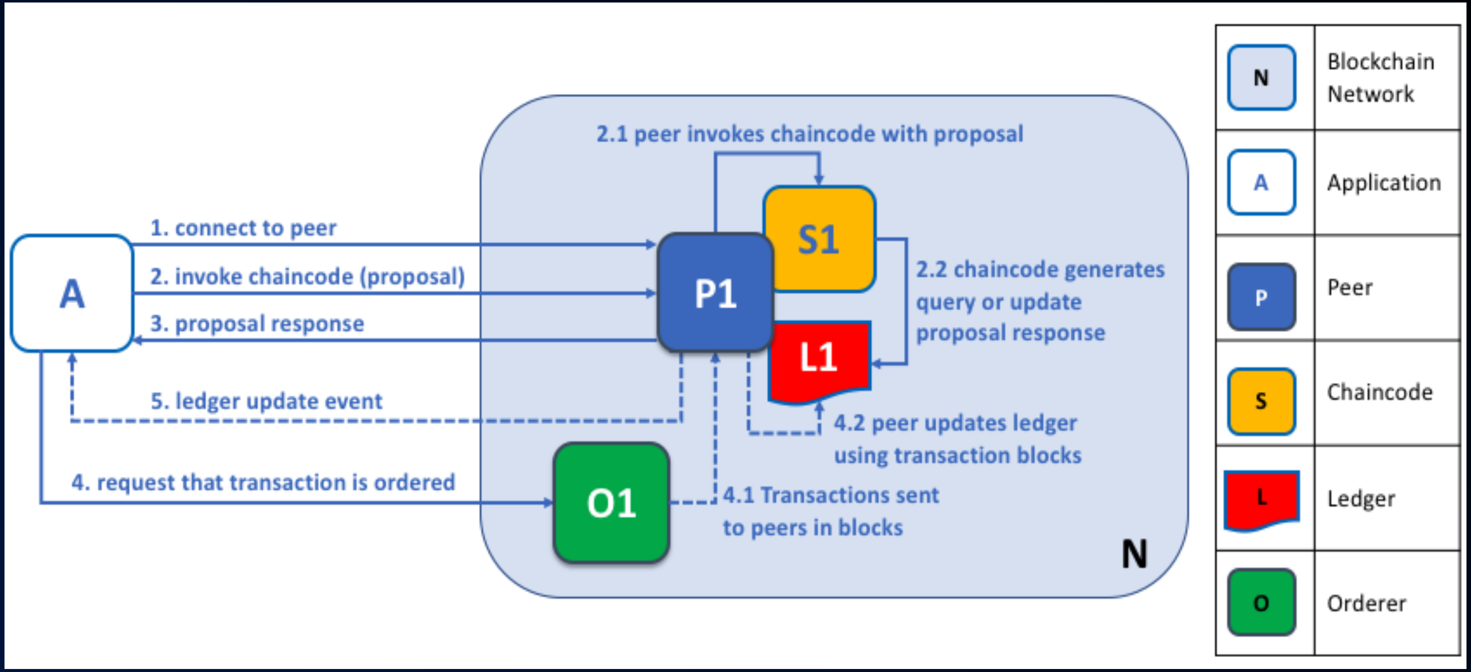
Consensus

- It is the process to keep Ledger transactions synchronized
    - Only when the transactions are approved by the proper participant
    - Ensuring that the Ledger are updated with the same transactions in the same order



Transaction Flow

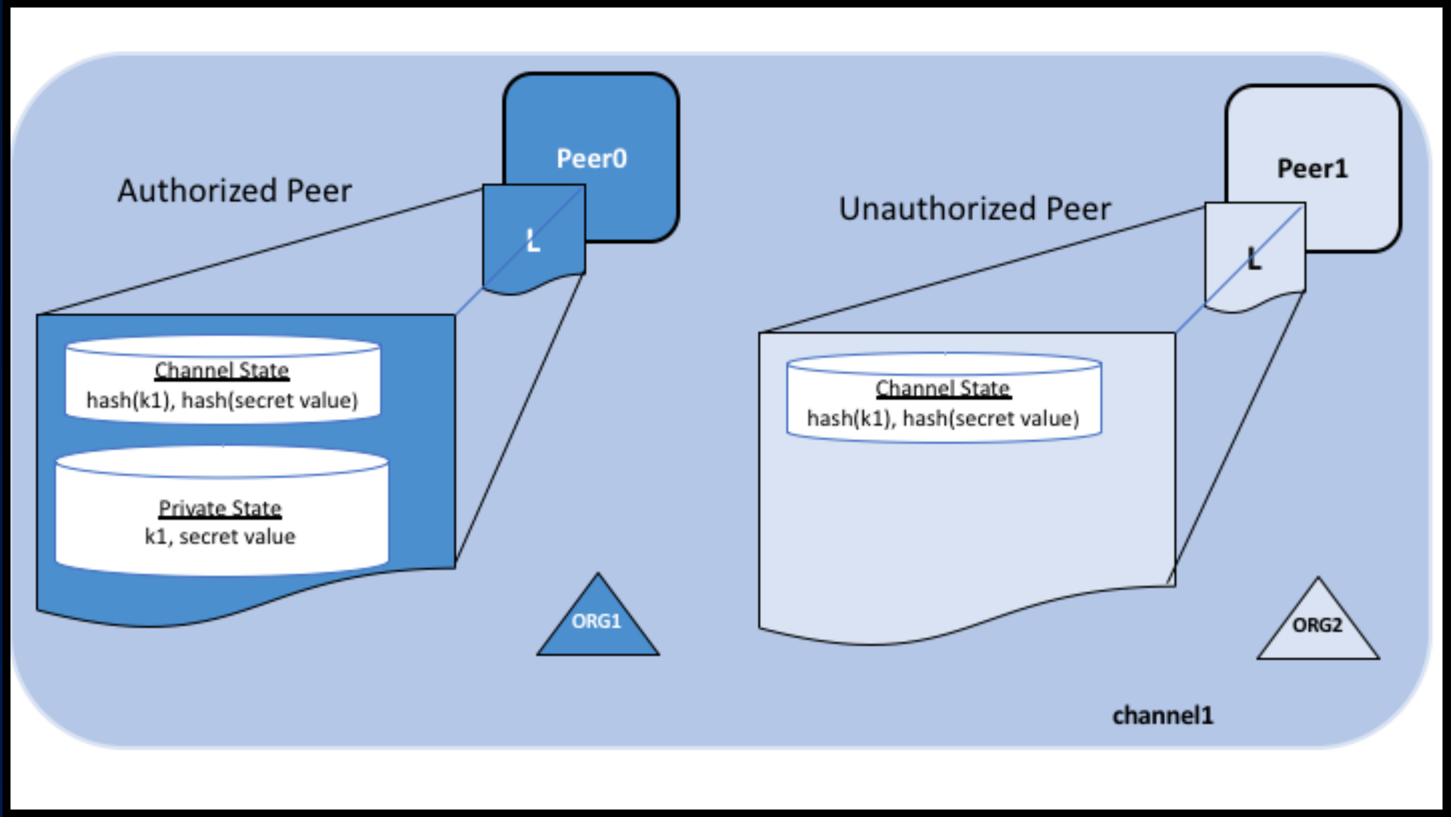
a. One client application A submits to an endorsing peer P1 an operation O to the Ledger (1.,2.)
b. P1 does not reply immediately, as well as does not apply immediately the result of O to the Ledger, because it simulates such operations and waits for the consensus from other endorsing peers. (2.1,2.2)


2.1 peer invokes chaincode with proposal
1. connect to peer
2. invoke chaincode (proposal)
3. proposal response
5. ledger update event
4. request that transaction is ordered
A
P1
S1
L1
O1
2.2 chaincode generates query or update proposal response
4.2 peer updates ledger using transaction blocks
4.1 Transactions sent to peers in blocks
N
N Blockchain Network
A Application
P Peer
S Chaincode
L Ledger
O Orderer

c. If the consensus is not met, an error is returned to A. Otherwise A receives all the transaction proposals from the endorsing peers signed by them. (3.)
d. A forwards the proposal to the orderers (4.)
e. The leading orderer creates blocks which are disseminated to the leading peers. (4.1)
f. Once a peer's Ledger is updated, the same peer sends an event to A stating that the operations O has been successfully applied to the Ledger (4.2, 5.)

Private Data

- Hyperledger Fabric provides a way to keep data secret between subset of organizations
- This is possible thanks to the Private Collections.
- …

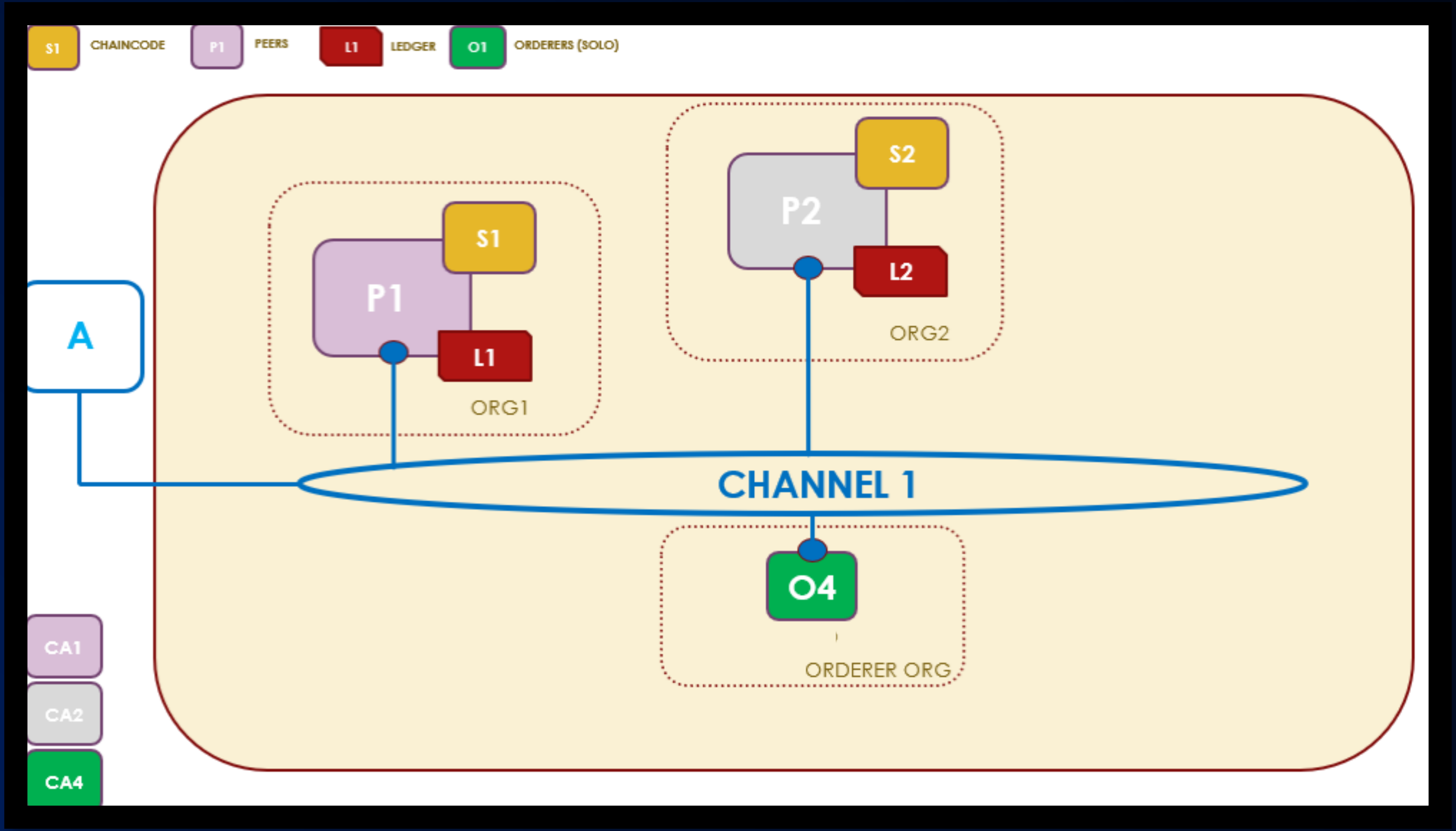


Hands-on - Prerequisites

- 1 VM with Ubuntu (server or desktop version (the desktop one is suggested)) with installed:
  - Docker and docker-compose:
    - `sudo apt-get -y install docker-compose`
    - Ensuring that the user belongs to the docker group:
      - a. `sudo usermod -aG docker <your_username>`
      - b. `sudo chown root:docker /var/run/docker.sock`
      - c. `sudo chown -R root:docker /var/run/docker`
      - d. relogin from the VM
  - `gradle`
  - `Java 11`
  - `jq`
  - `curl`
  - `git`
  - fabric images + binaries + directory of fabric-samples:
    - `curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh`
    - `./install-fabric.sh --fabric-version 2.5.9 --ca-version 1.5.12 docker binary samples`

Hands-on

1. Instantiate the network (channel)
2. Deploy chaincode
3. Run application
4. …



Hands-on - Chaincode asset-transfer-basic

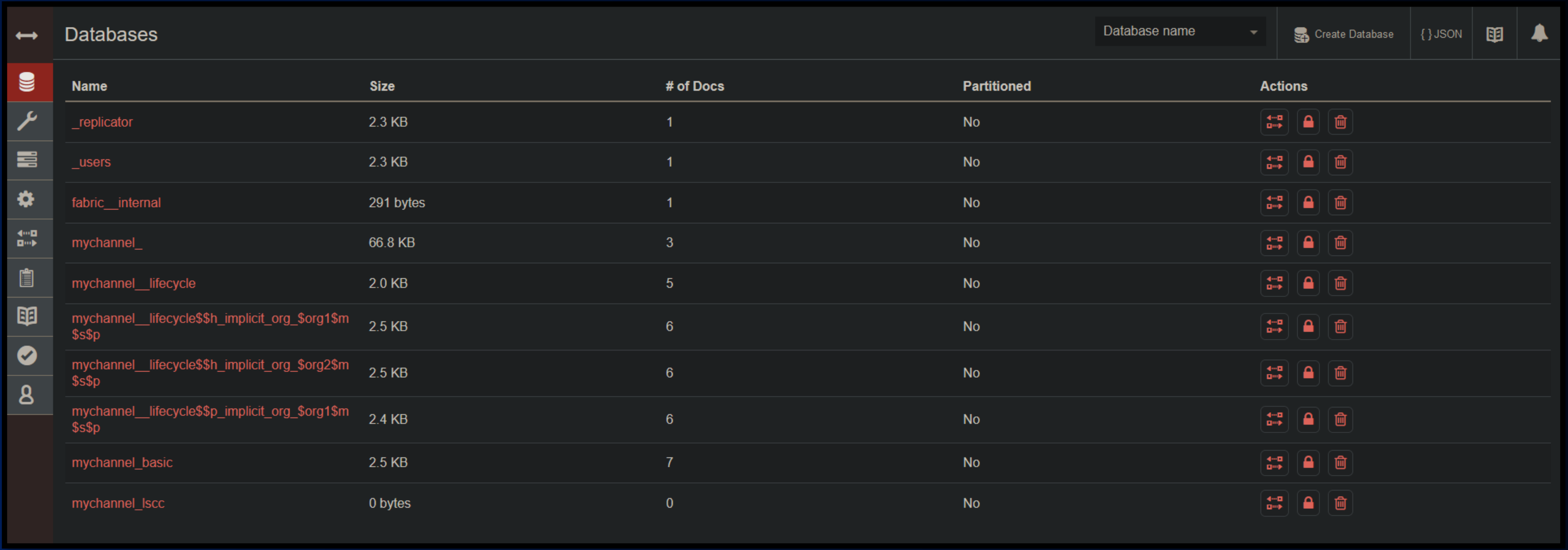
- **`InitLedger`**`()`
- **`CreateAsset`**`(assetID, color, size, owner, appraisedValue)`
- **`ReadAsset`**`(assetID)`
- **`UpdateAsset`**`(assetID, color, size, owner, appraisedValue)`
- **`DeleteAsset`**`(assetID)`
- **`AssetExists`**`(assetID)`
- **`TransferAsset`**`(assetID, newOwner)`
- **`GetAllAssets`**`()`

Hands-on - Application asset-transfer-basic

1. Create connection thanks to peer certificate, peer endpoint, peer name, organization’s MSP id + user certificate (client identity) and user private key (signer),
2. It initializes the Ledger,
3. It shows the assets just created,
4. It creates an asset with `id=asset<timestamp>`,
5. It transfers the ownership of asset with `id=asset<timestamp>` from Tom to Saptha,
6. It shows the value of the asset with `id=asset<timestamp>`,
7. It tries to update and asset with `id=asset70`.

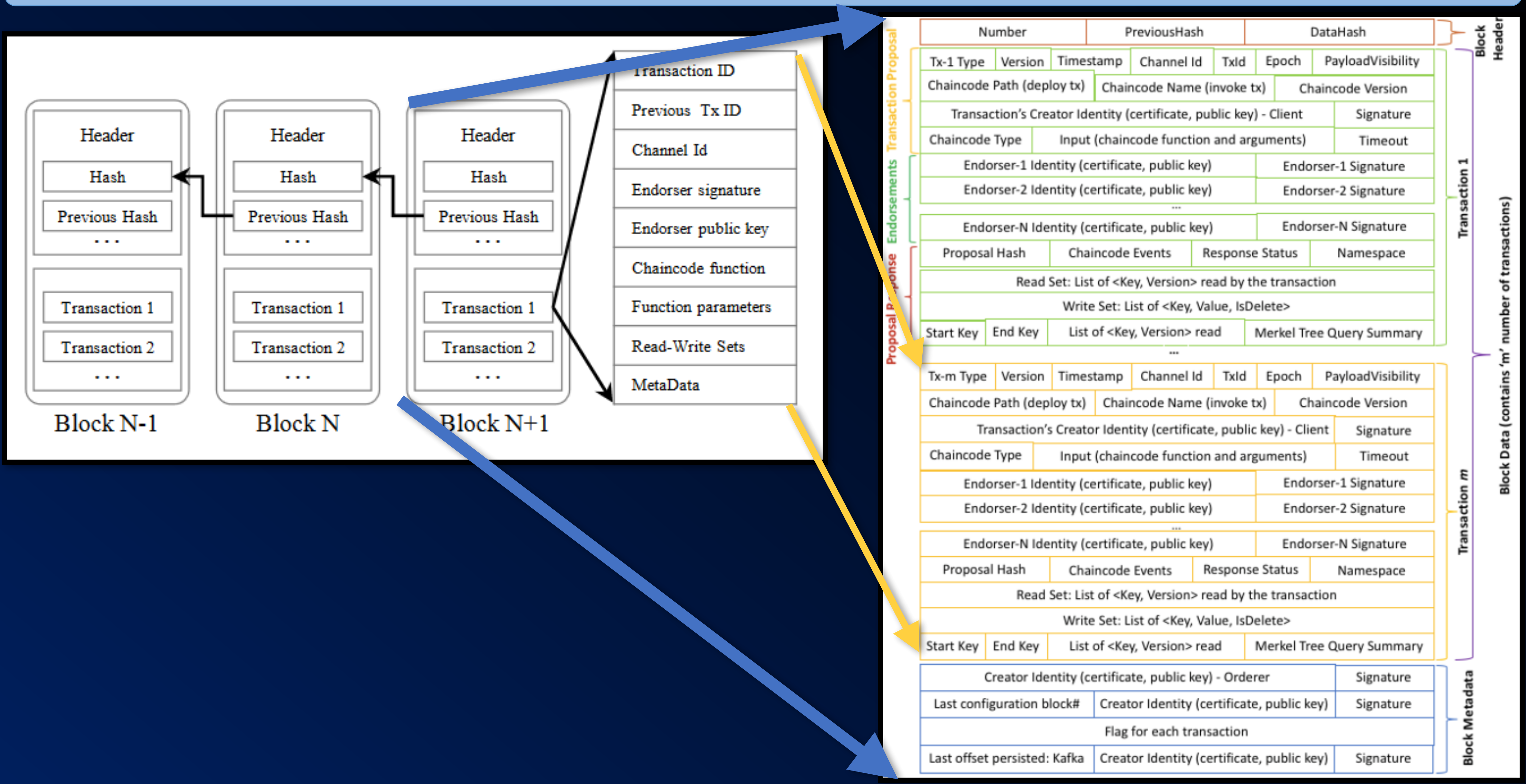
Hands-on - World State (CouchDB)

- `http://<vm_address>:5984/_utils` (org1)
- `http://<vm_address>:7984/_utils` (org2)
- Credentials: user: `admin`, password: `adminpw`



Name	Size	# of Docs	Partitioned	Actions
_replicator	2.3 KB	1	No	
_users	2.3 KB	1	No	
fabric__internal	291 bytes	1	No	
mychannel_	66.8 KB	3	No	
mychannel__lifecycle	2.0 KB	5	No	
mychannel__lifecycle$$h_implicit_org_$org1$msp	2.5 KB	6	No	
mychannel__lifecycle$$h_implicit_org_$org2$msp	2.5 KB	6	No	
mychannel__lifecycle$$p_implicit_org_$org1$msp	2.4 KB	6	No	
mychannel_basic	2.5 KB	7	No	
mychannel_lscc	0 bytes	0	No	

Hands-on - Blockchain


Header
Hash
Previous Hash
Transaction 1
Transaction 2
Block N-1
Block N
Block N+1
Transaction ID
Previous Tx ID
Channel Id
Endorser signature
Endorser public key
Chaincode function
Function parameters
Read-Write Sets
MetaData
Number
PreviousHash
DataHash
Block Header
Transaction Proposal
Tx-1 Type
Version
Timestamp
Channel Id
TxId
Epoch
PayloadVisibility
Chaincode Path (deploy tx)
Chaincode Name (invoke tx)
Chaincode Version
Transaction's Creator Identity (certificate, public key) - Client
Signature
Chaincode Type
Input (chaincode function and arguments)
Timeout
Endorsements
Endorser-1 Identity (certificate, public key)
Endorser-1 Signature
Endorser-2 Identity (certificate, public key)
Endorser-2 Signature
Endorser-N Identity (certificate, public key)
Endorser-N Signature
Proposal Response
Proposal Hash
Chaincode Events
Response Status
Namespace
Read Set: List of <Key, Version> read by the transaction
Write Set: List of <Key, Value, IsDelete>
Start Key
End Key
List of <Key, Version> read
Merkel Tree Query Summary
Transaction 1
Tx-m Type
Transaction m
Block Data (contains 'm' number of transactions)
Creator Identity (certificate, public key) - Orderer
Last configuration block#
Creator Identity (certificate, public key)
Flag for each transaction
Last offset persisted: Kafka
Block Metadata

References and some images taken from…

- https://hyperledger-fabric.readthedocs.io/en/latest/
  - And internals…
- https://hyperledger-fabric.readthedocs.io/en/release-2.2/
  - And internals…
- E. Zhou, H. Sun, B. Pi, J. Sun, K. Yamashita, Y. Nomura “Ledgerdata Refiner: A Powerful Ledger Data Query Platform for Hyperledger Fabric”
- The ones at slide on “Use Cases”