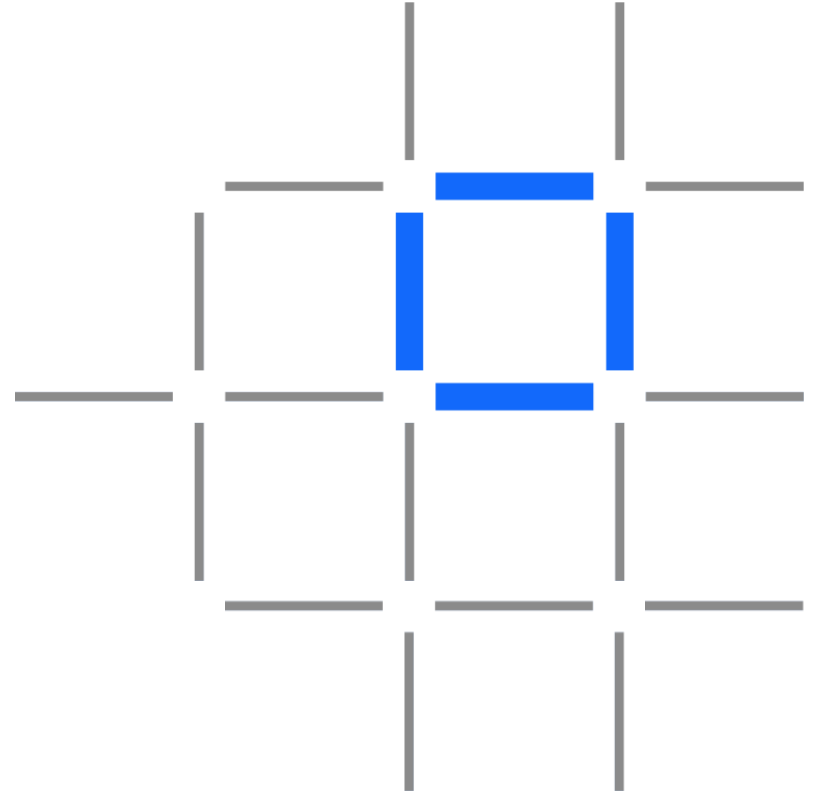


Module 1b

A Technical Introduction to Blockchain



IBM Blockchain



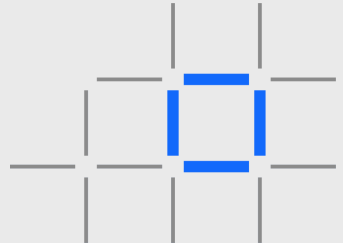
Understanding
Hyperledger Fabric



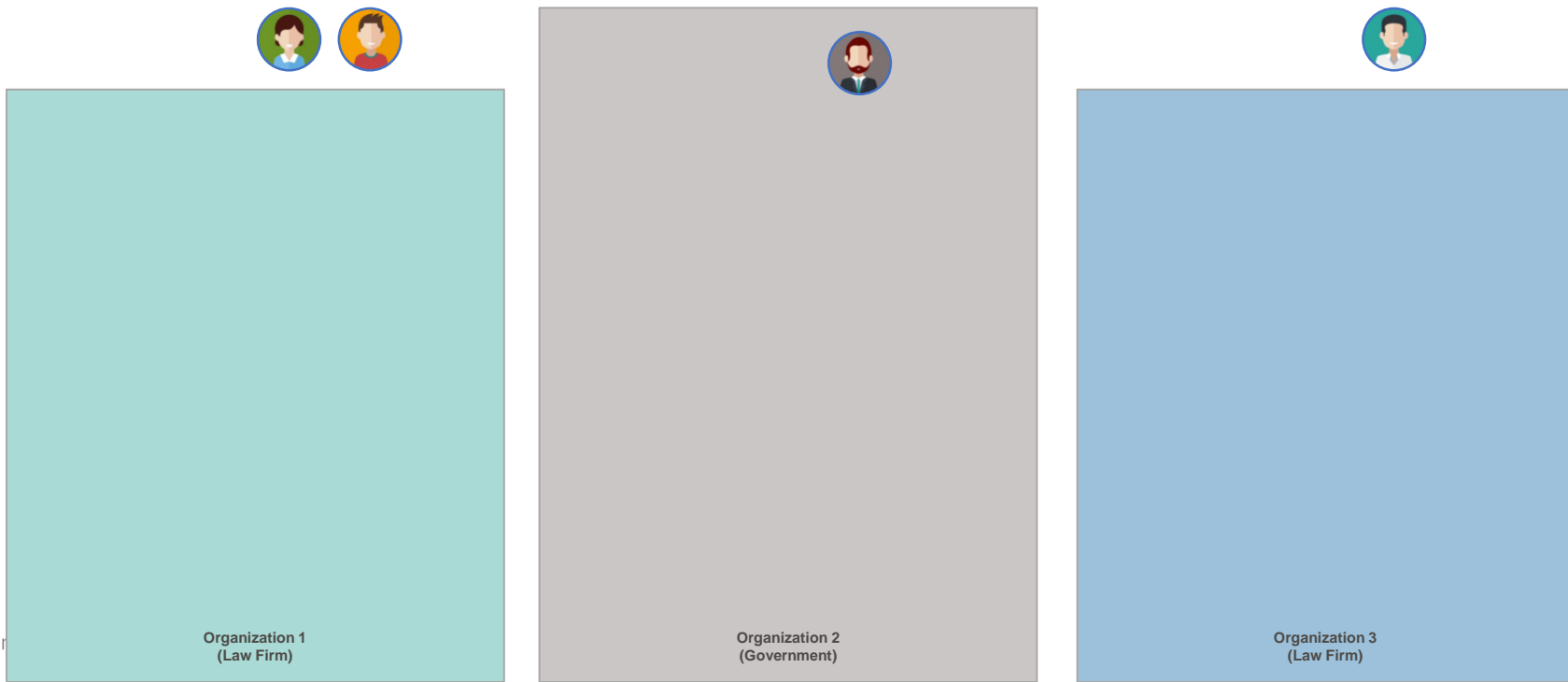
Transaction Flow in
Hyperledger Fabric



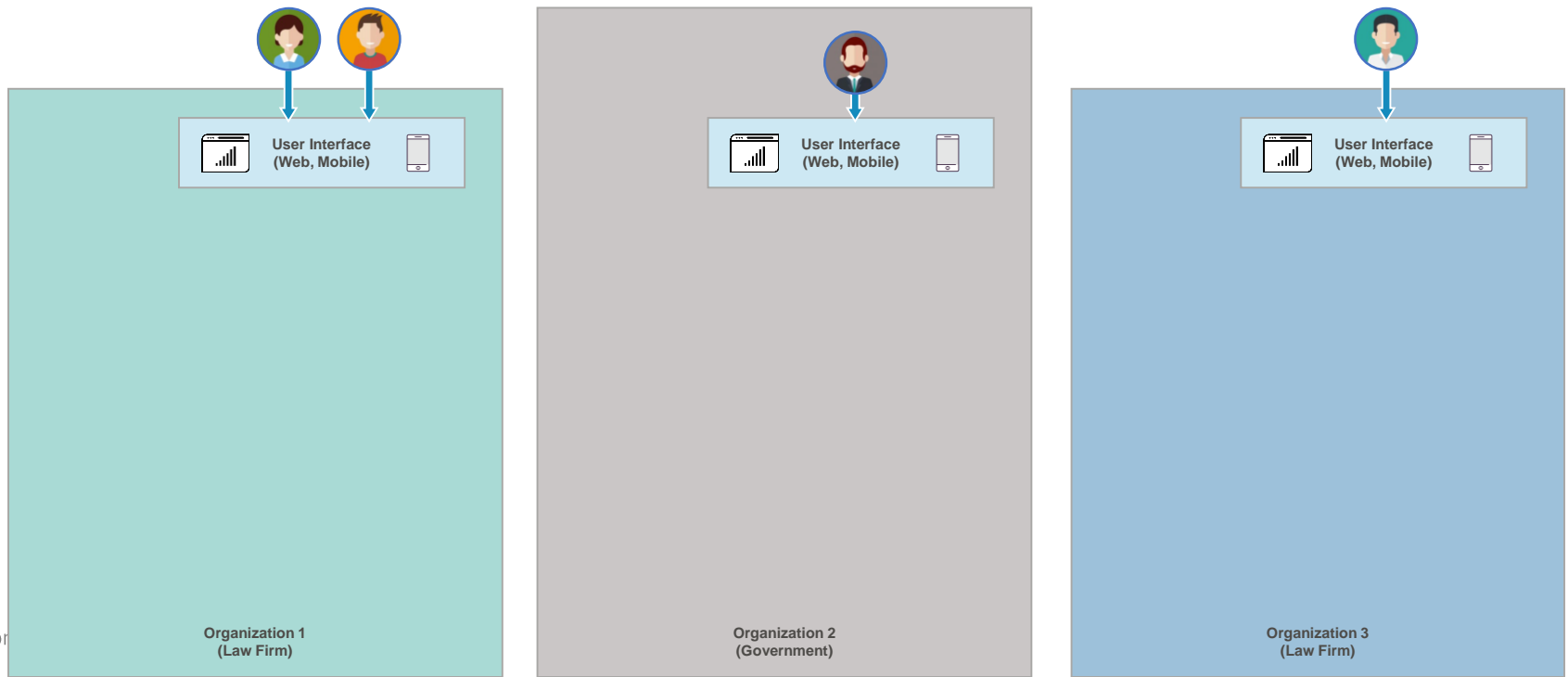
Who is involved in a
Hyperledger Fabric project?



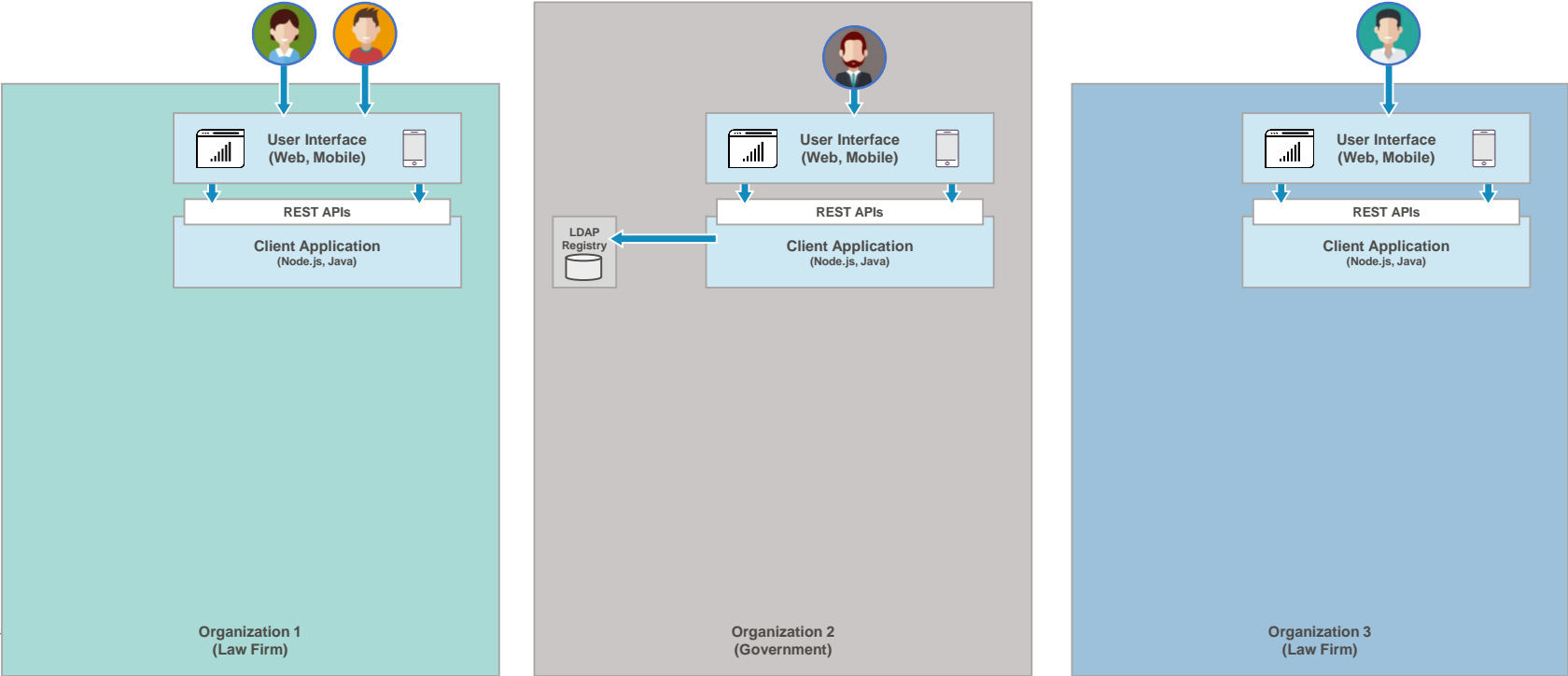
Blockchain is a team sport, so it's important to begin with the perspective of multiple organizations that want to work together, and the individuals who will be interacting with the blockchain-based solution via those organizations. This is a **business network**.



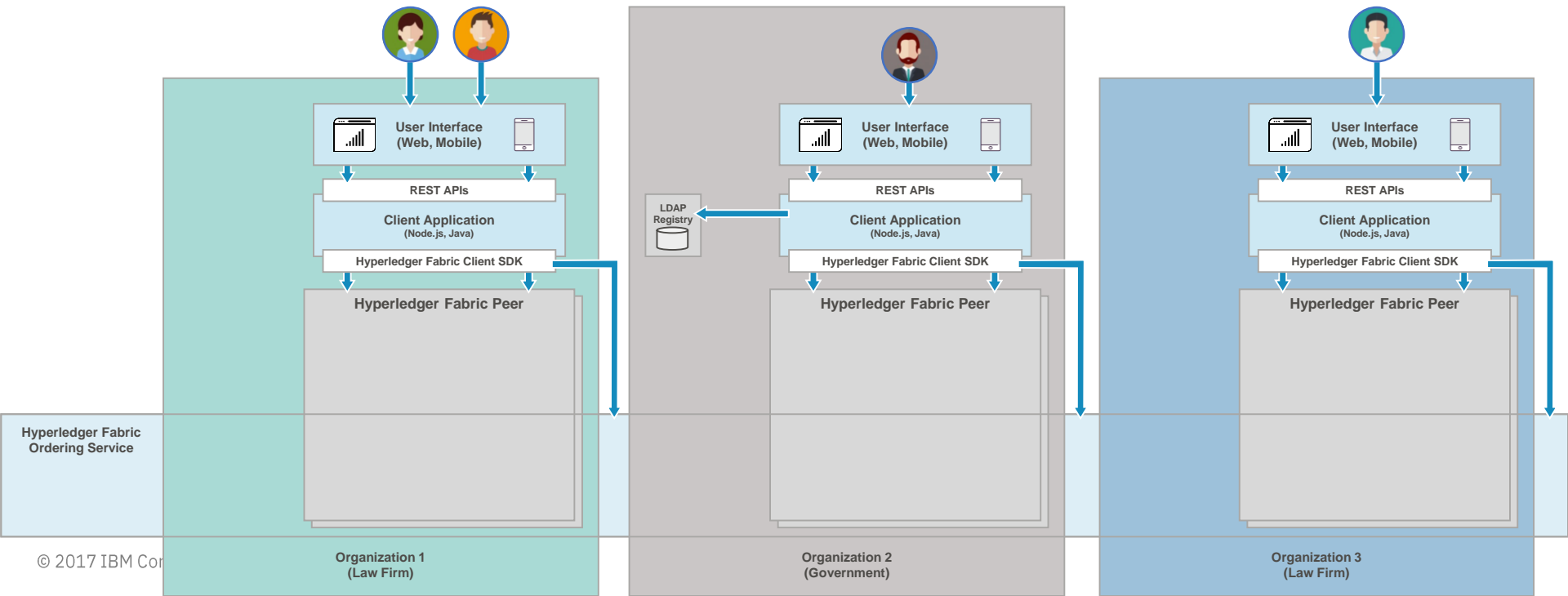
Individuals interact with public-facing and organization-internal applications, through a variety of user interfaces (mobile, web, etc.)



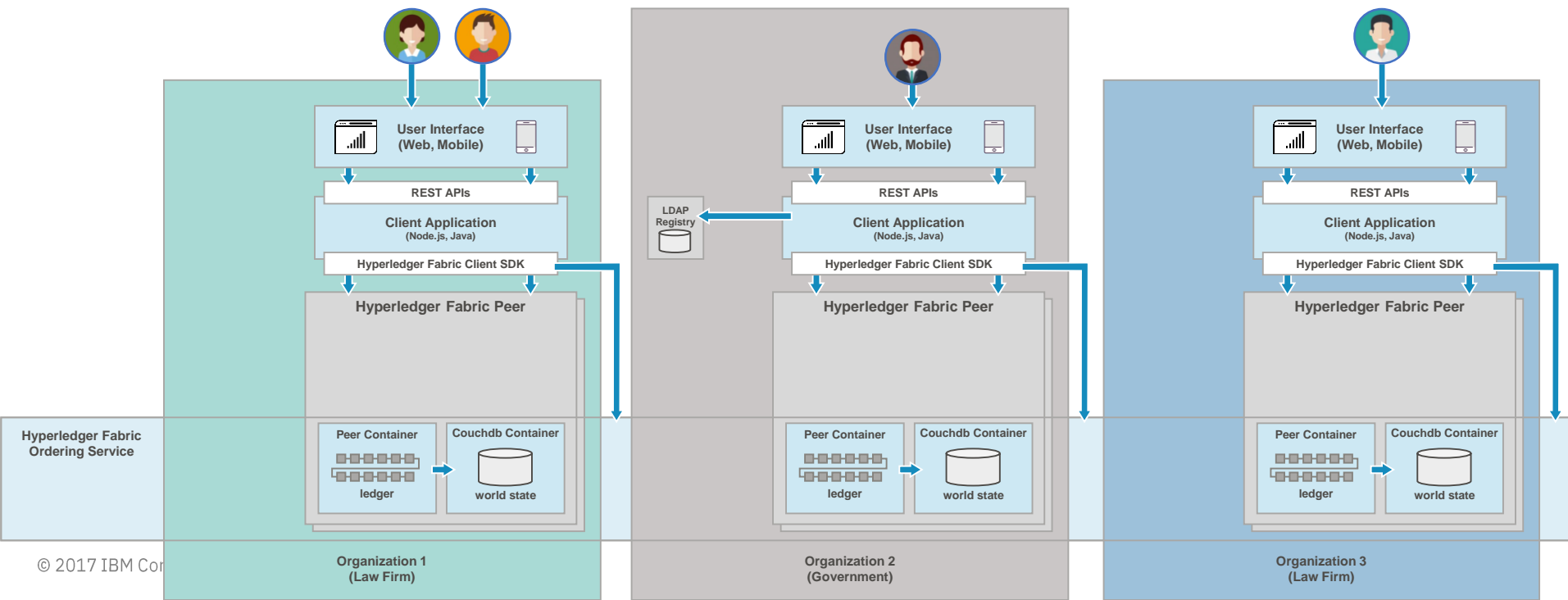
User interfaces typically communicate with backend applications through REST APIs which expose functionality to the UI layer according to the business domain of the application. Applications have various ways of authenticating users (e.g. LDAP, SSO, OAUTH, etc.)



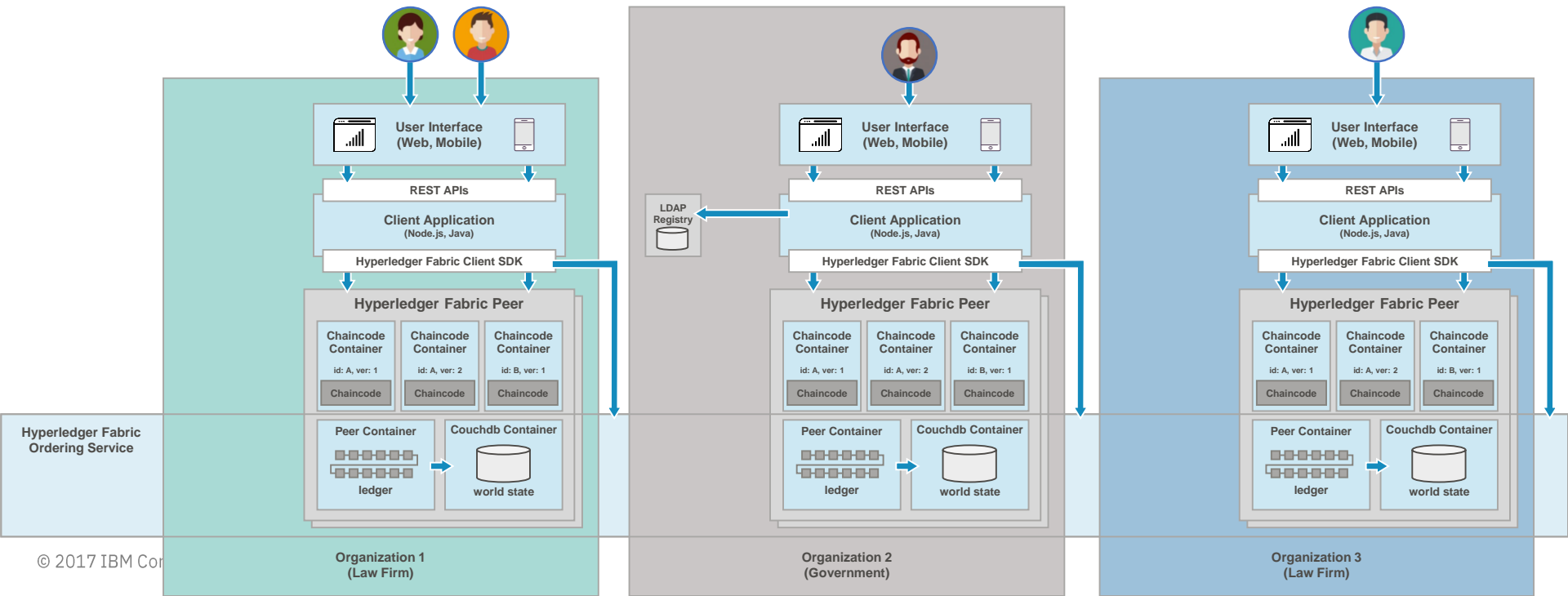
In a Hyperledger Fabric application, the application layer acts as a **blockchain client**, in that it connects to the Fabric components (using the Hyperledger Fabric Client SDK) on behalf of the user in order to query the ledger and execute transactions.



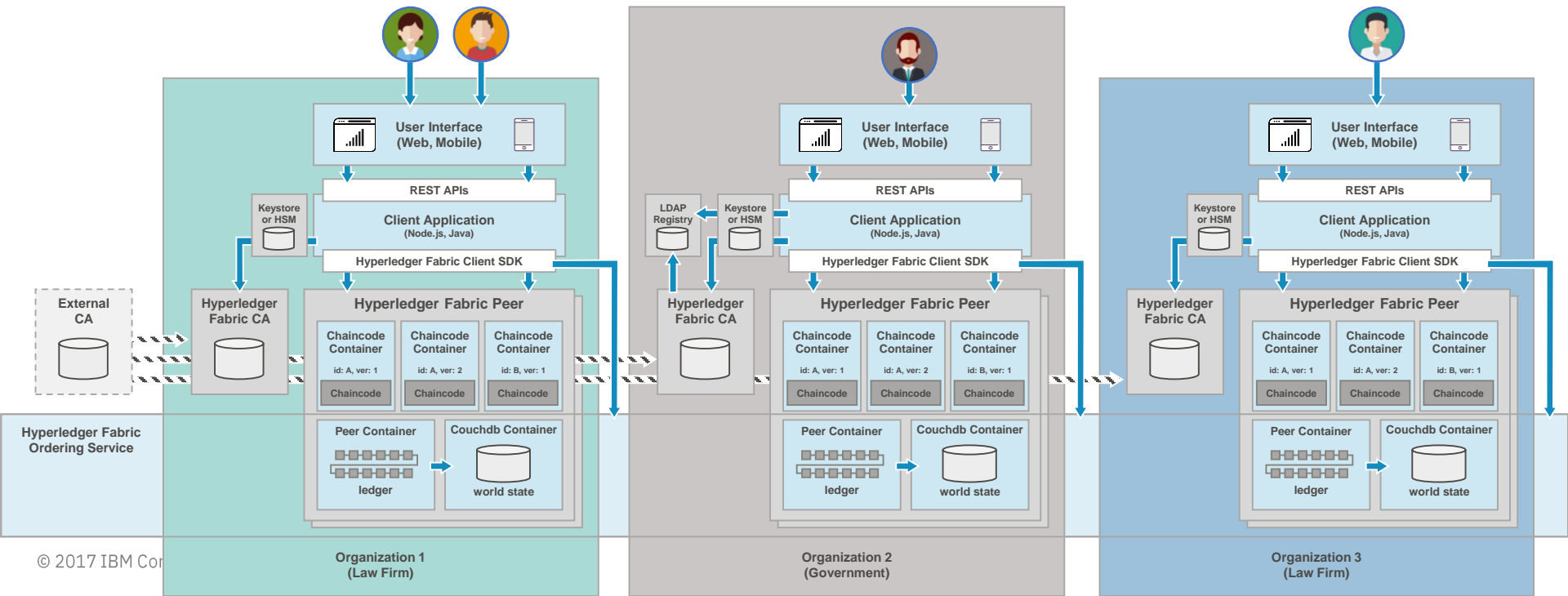
Peers are a primary concept in Hyperledger Fabric. A peer is a Fabric application component that runs in a Docker container and is responsible for maintaining a copy of the ledger, and providing programmatic access to the information on the ledger via the world state DB. An organization will typically have more than one peer, primarily for high availability.



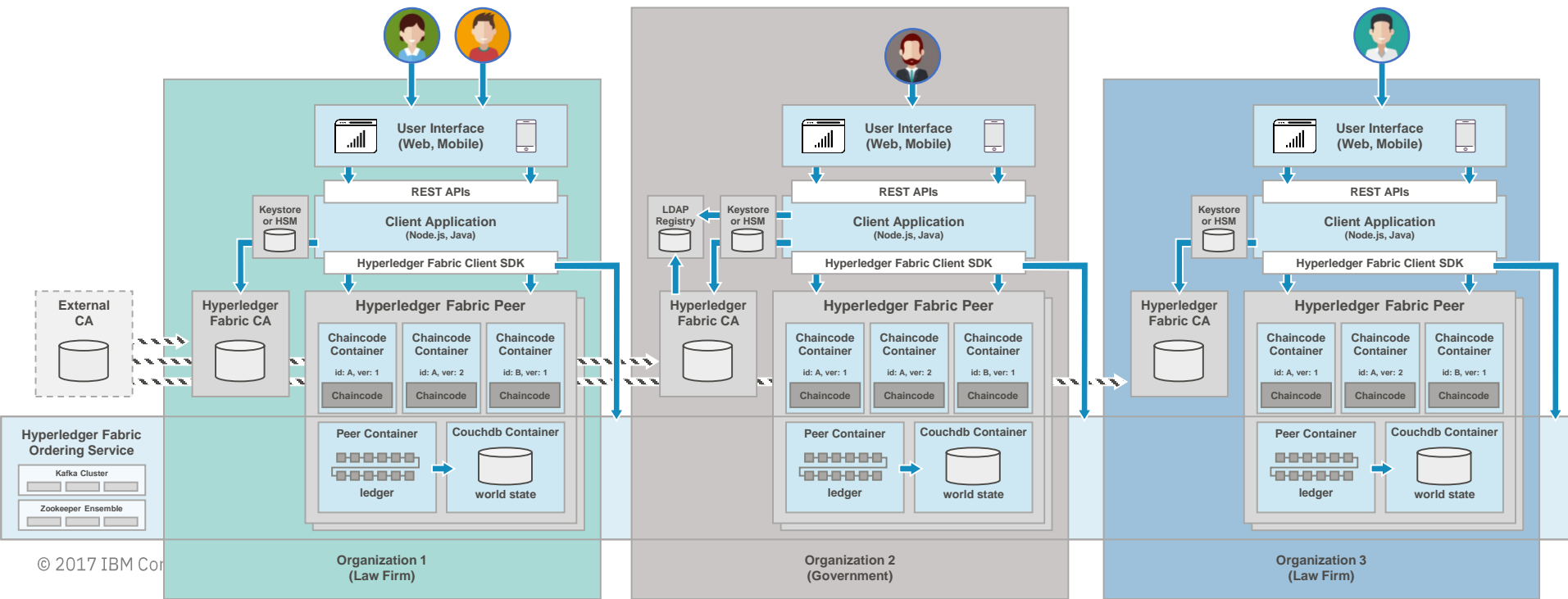
Peers perform application-specific function by executing smart contracts, or more technically, chaincode. Chaincode is managed application code with a well-defined lifecycle that runs within a peer. Chaincode uses Hyperledger Fabric APIs to interact with the world state.



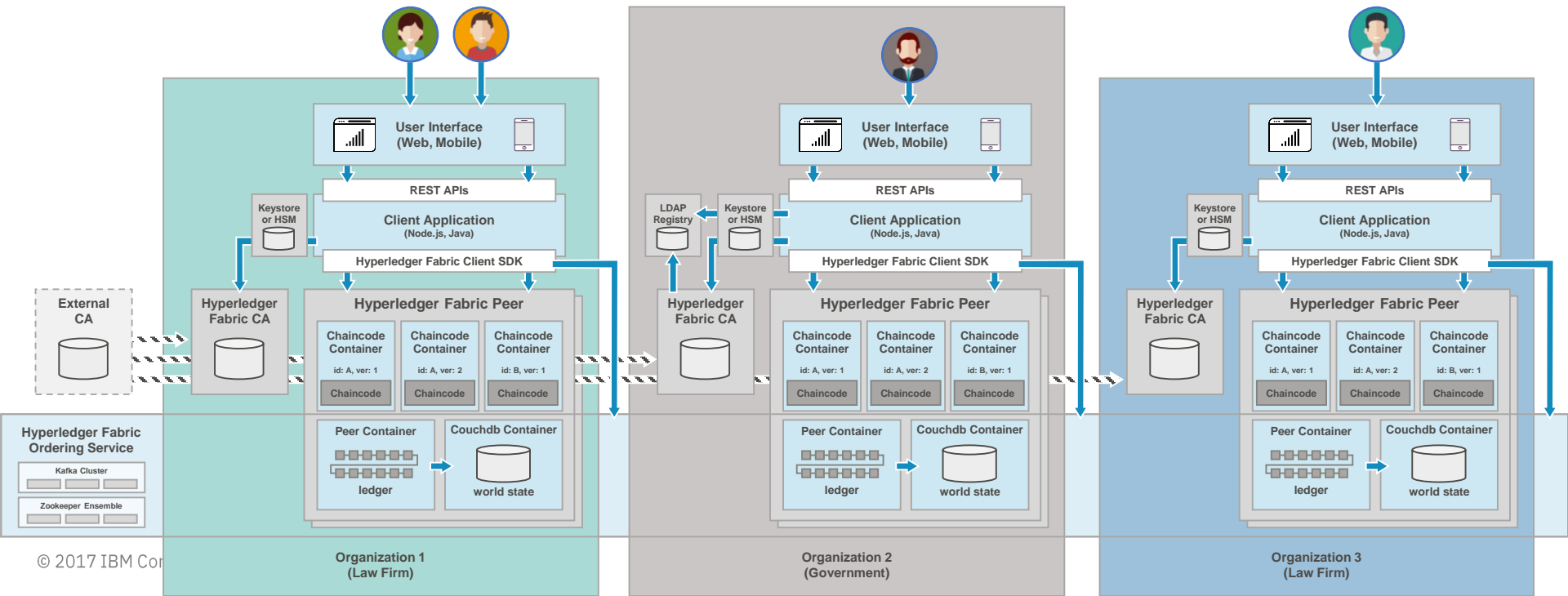
Because Hyperledger Fabric is a private, permissioned blockchain, identity plays a critical role. Identity of individuals as well as Fabric Components such as peers and orderers are defined by certificates, issued by a **certificate authority**. Any certificate authority can be (and often is) used, but Fabric also includes one in the box, called the Hyperledger Fabric CA.



The Ordering Service is the network-level service responsible for determining the order in which transactions coming from any participant in the business network should be committed to the ledger. For high availability and throughput, a Kafka cluster backed by a Zookeeper ensemble is used to distribute blocks to peers (i.e. perform **consensus**).

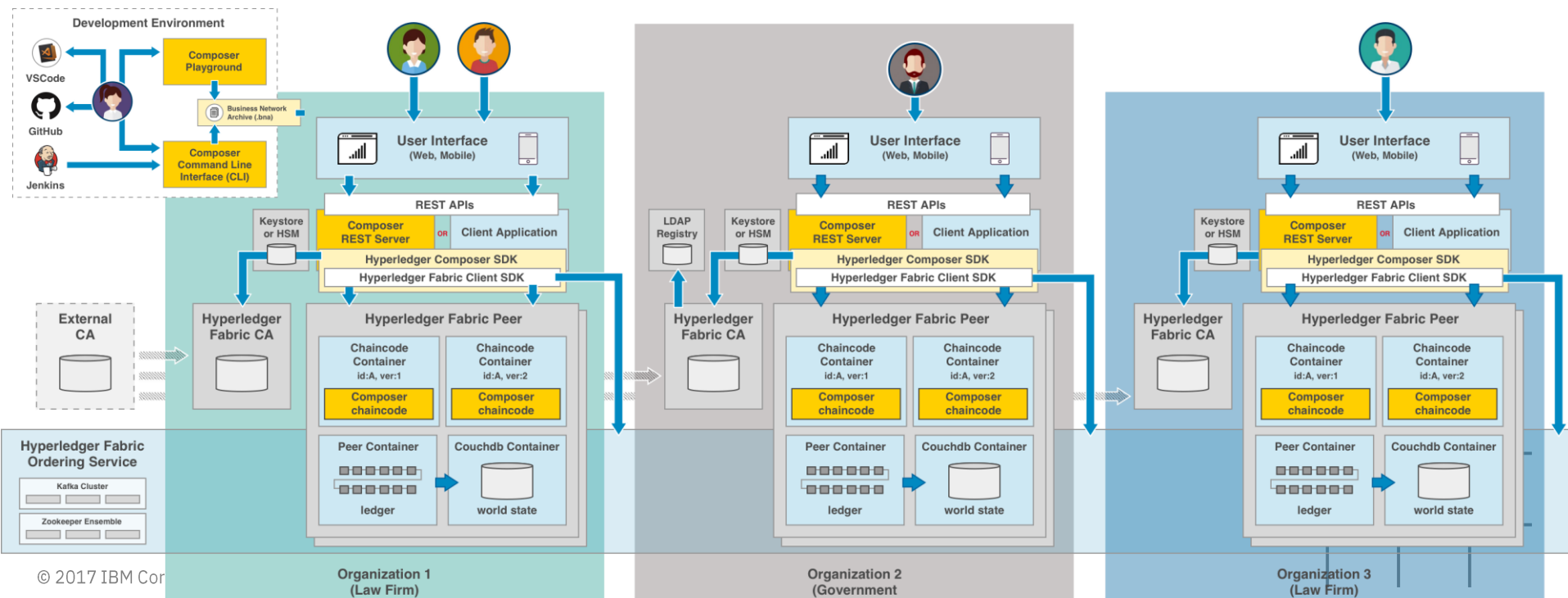


This makes up our final conceptual architecture diagram for a Hyperledger Fabric application



Hyperledger Fabric + Composer

Hyperledger Composer adds a number of components that leverage the underlying Fabric concepts and infrastructure, simplifying the programming model





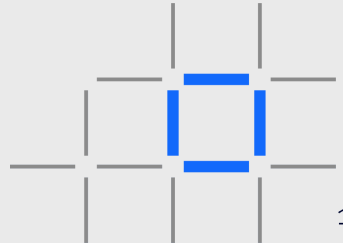
Understanding
Hyperledger Fabric






Transaction Flow in
Hyperledger Fabric



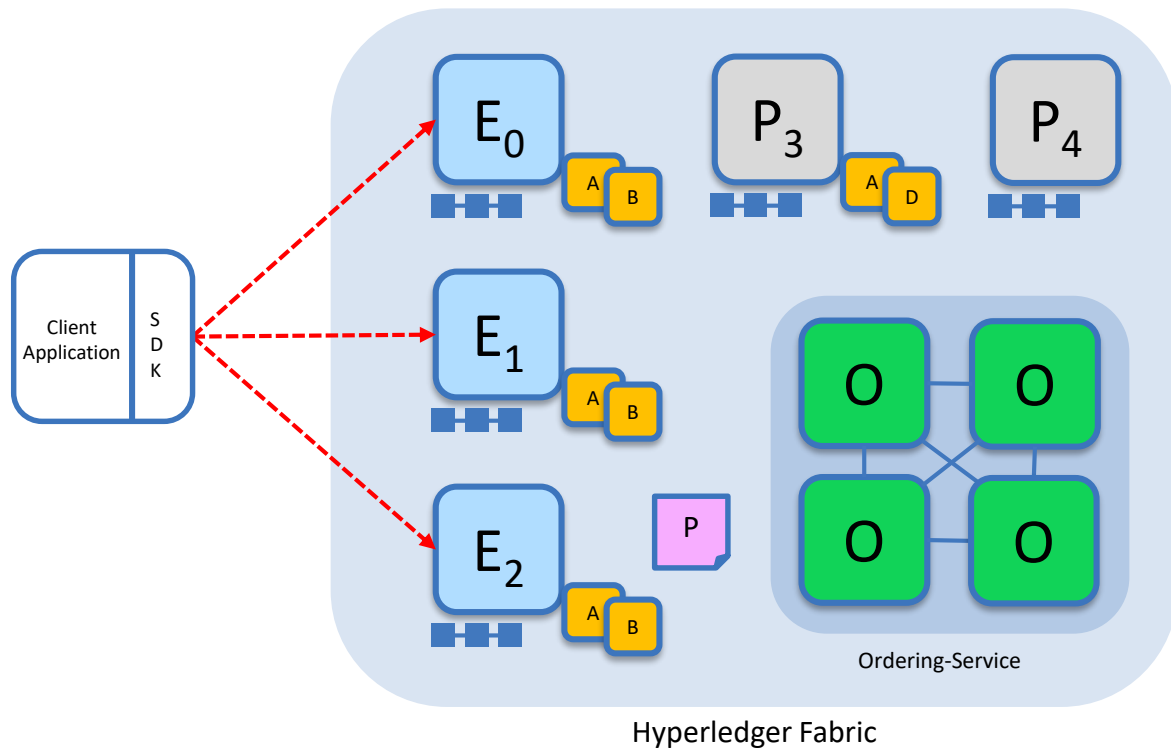
Who is involved in a
Hyperledger Fabric project?



Transaction Flow in Hyperledger Fabric

	Committing Peer: Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	Endorsing Peer: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
	Orderer Node (part of the Ordering Service): 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.

Sample transaction: Step 1/7 – Propose transaction



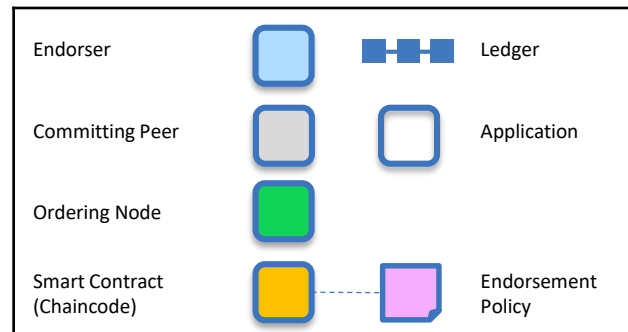
Application proposes transaction

Endorsement policy:

- “ E_0 , E_1 and E_2 must sign”
- (P_3 , P_4 are not part of the policy)

Client application submits a transaction proposal for Smart Contract A. It must target the required peers $\{E_0, E_1, E_2\}$

Key:



Sample transaction: Step 2/7 – Execute proposal

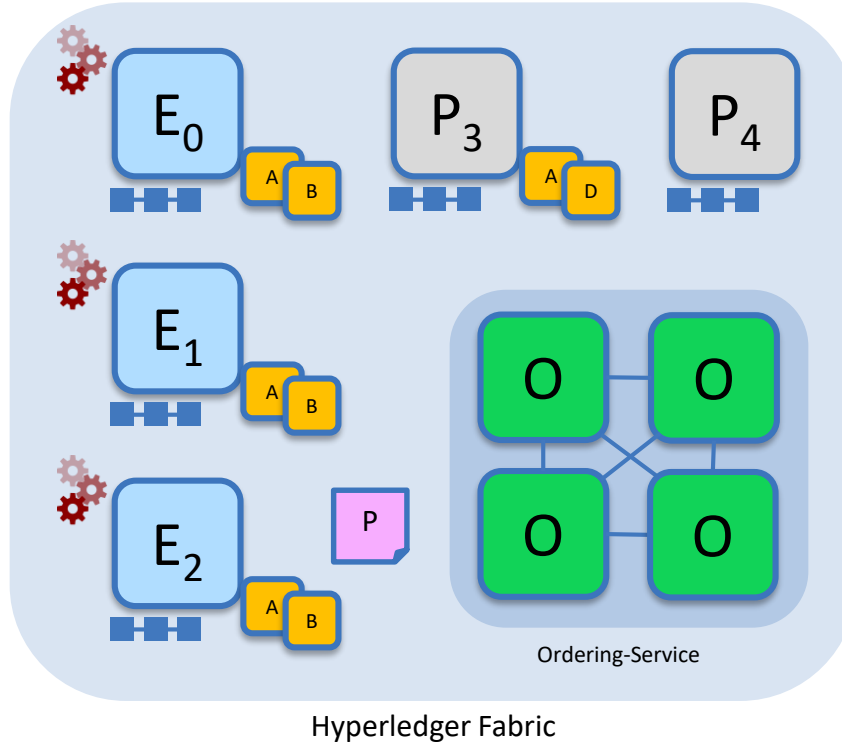
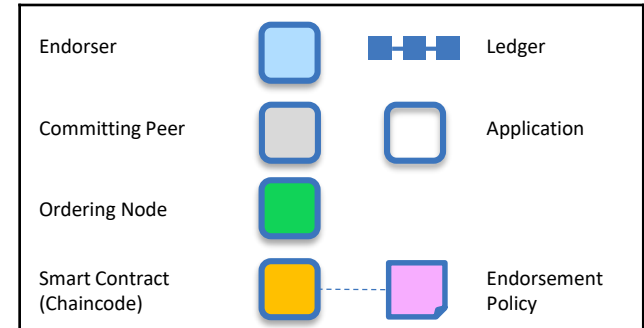
Endorsers Execute Proposals

E_0 , E_1 & E_2 will each execute the proposed transaction. None of these executions will update the ledger

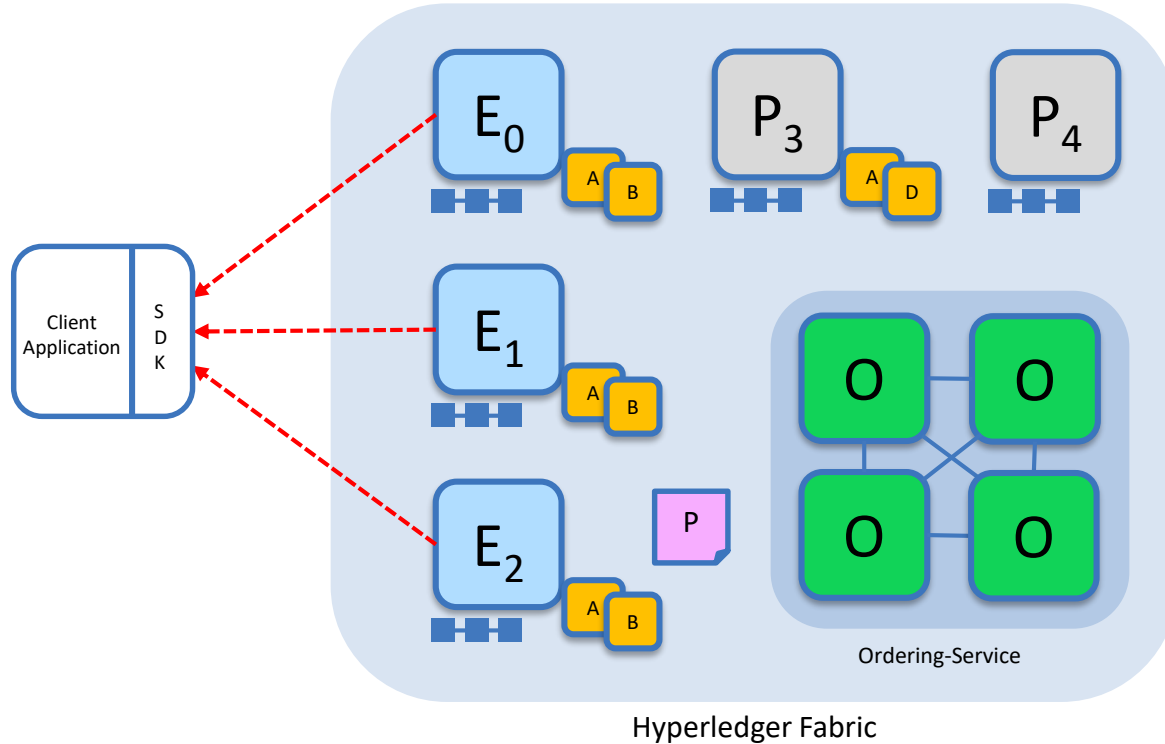
Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

Transactions can be signed & encrypted

Key:



Sample transaction: Step 3/7 – Proposal Response



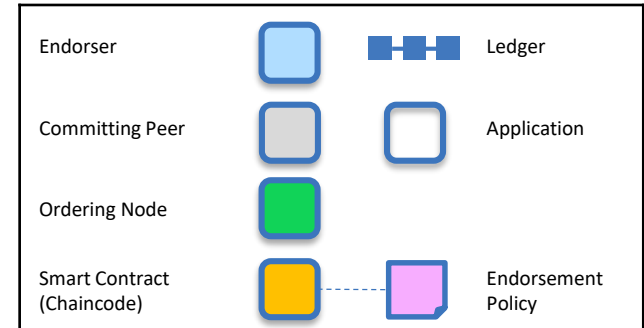
Application receives responses

RW sets are asynchronously returned to application

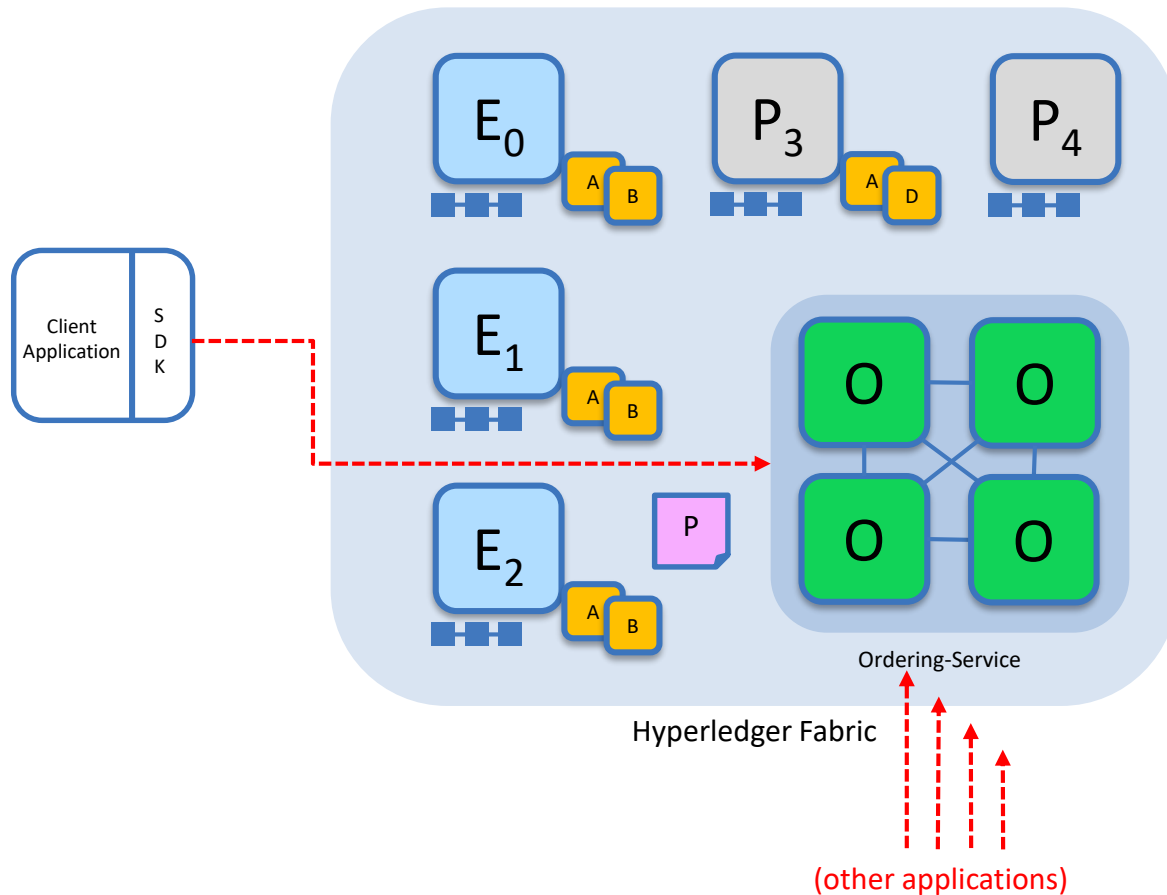
The RW sets are signed by each endorser, and also includes each record version number

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

Key:



Sample transaction: Step 4/7 – Order Transaction

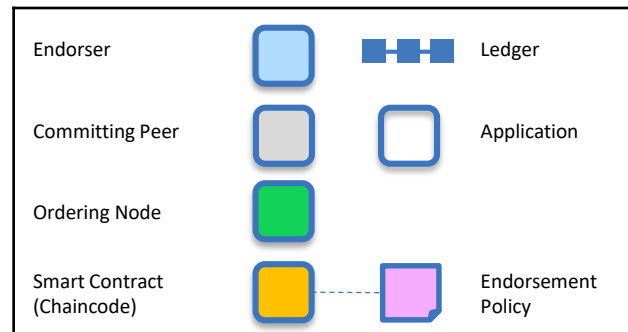


Application submits responses for ordering

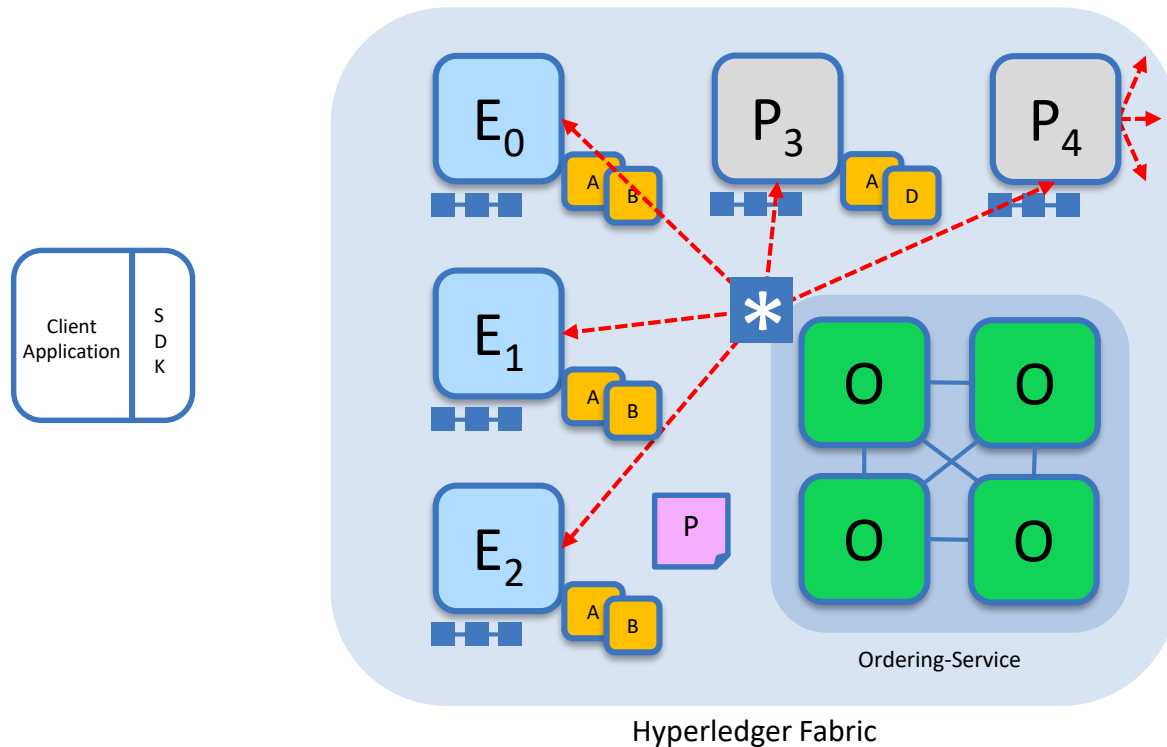
Application submits responses as a transaction to be ordered.

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

Key:



Sample transaction: Step 5/7 – Deliver Transaction



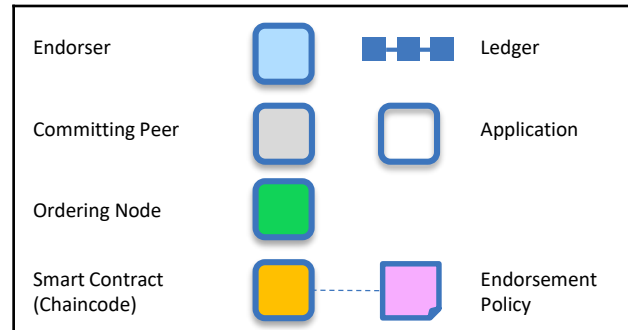
Orderer delivers to all committing peers

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 tolerance)

Key:



Sample transaction: Step 6/7 – Validate Transaction

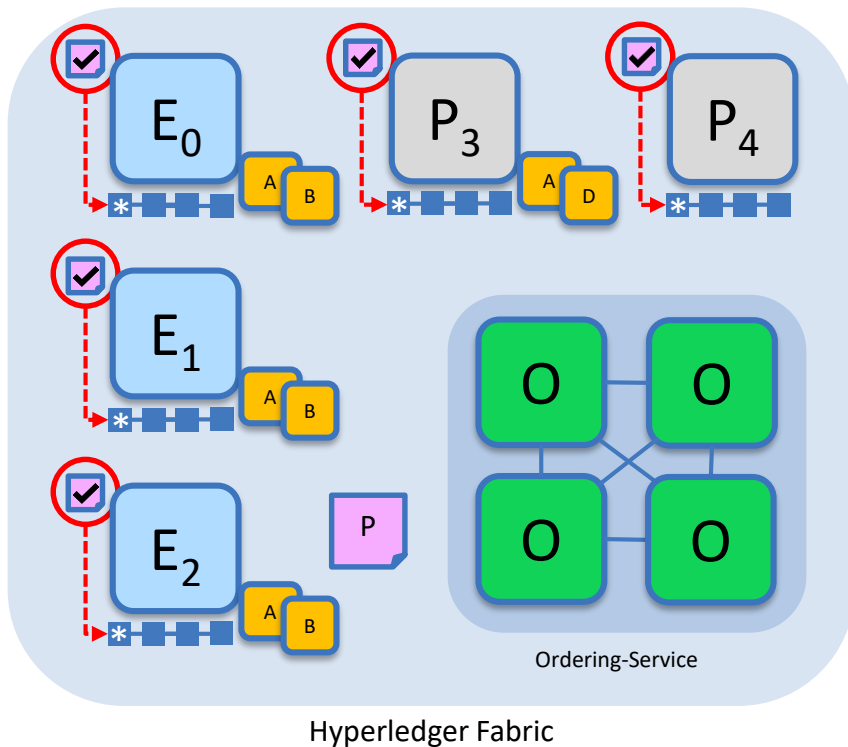
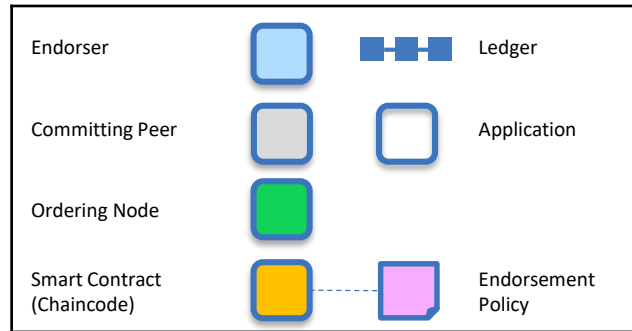
Committing peers validate transactions

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

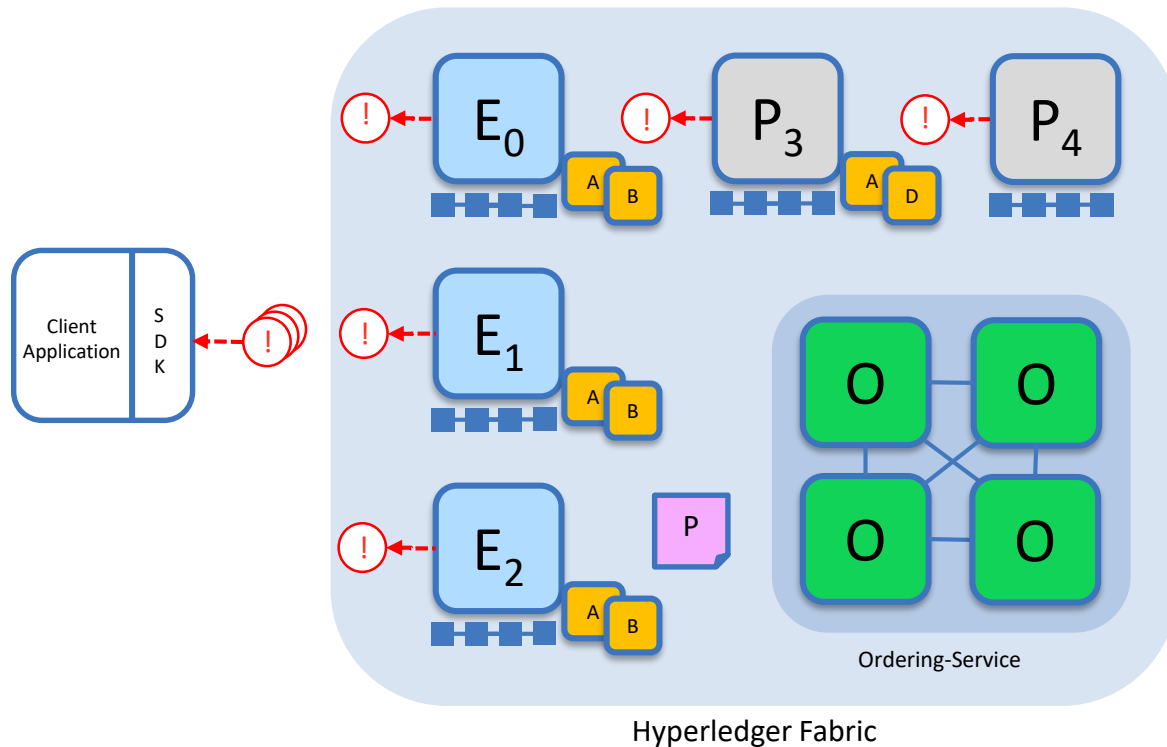
Validated transactions are applied to the world state and retained on the ledger

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

Key:



Sample transaction: Step 7/7 – Notify Transaction

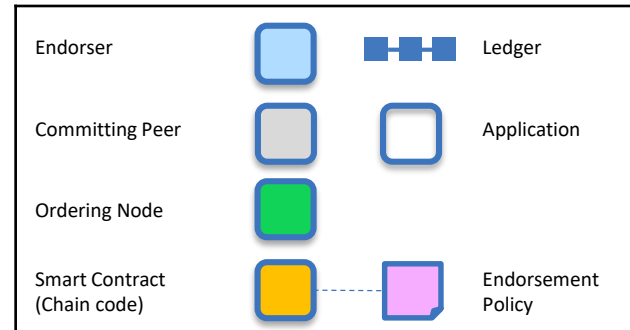


Committing peers notify applications

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

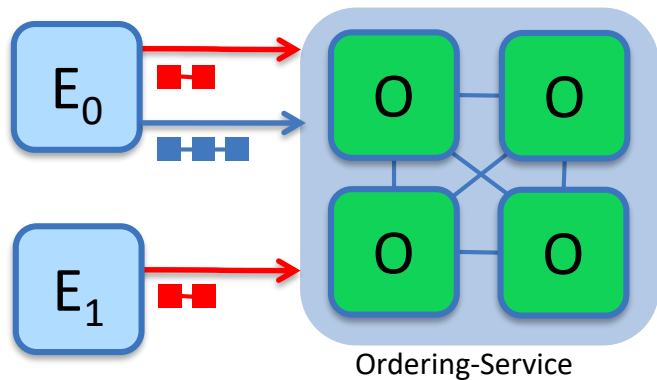
Applications will be notified by each peer to which they are connected

Key:



Channels

Channels are used to enforce data and business logic separation within a single Hyperledger Fabric business network



- Chaincode is installed on peers that need to access the worldstate
- Chaincode is instantiated on specific channels for specific peers
- Ledgers exist in the scope of a channel
 - Ledgers can be shared across an entire network of peers
 - Ledgers can be included only on a specific set of participants
- Peers can participate in multiple channels
- Concurrent execution for performance and scalability



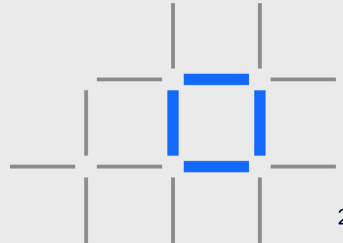
Understanding
Hyperledger Fabric



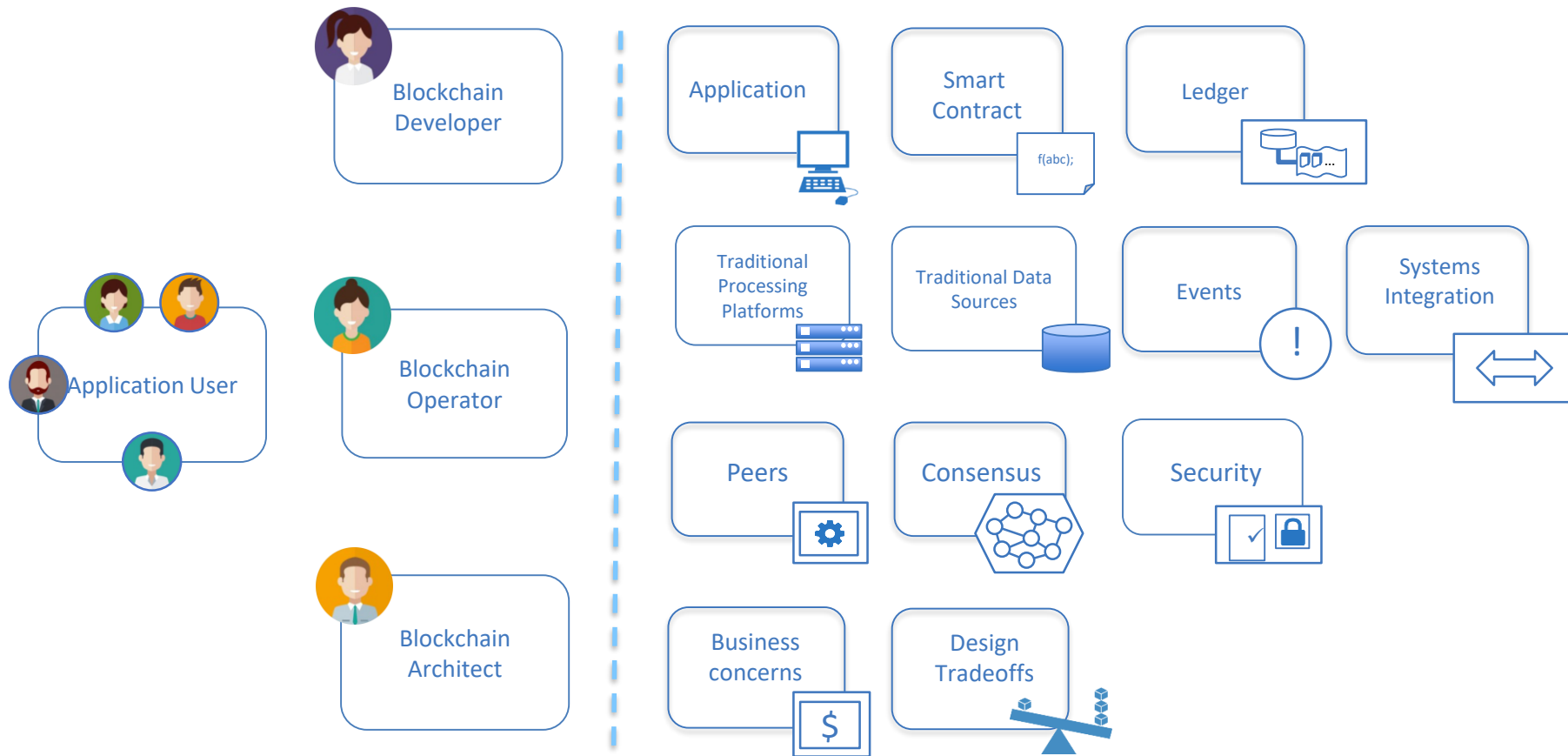
Transaction Flow in
Hyperledger Fabric



Who is involved in a
Hyperledger Fabric project?



Summary of key concepts



Thank you

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org

© Copyright IBM Corporation 2017. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represents only goals and objectives. IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

