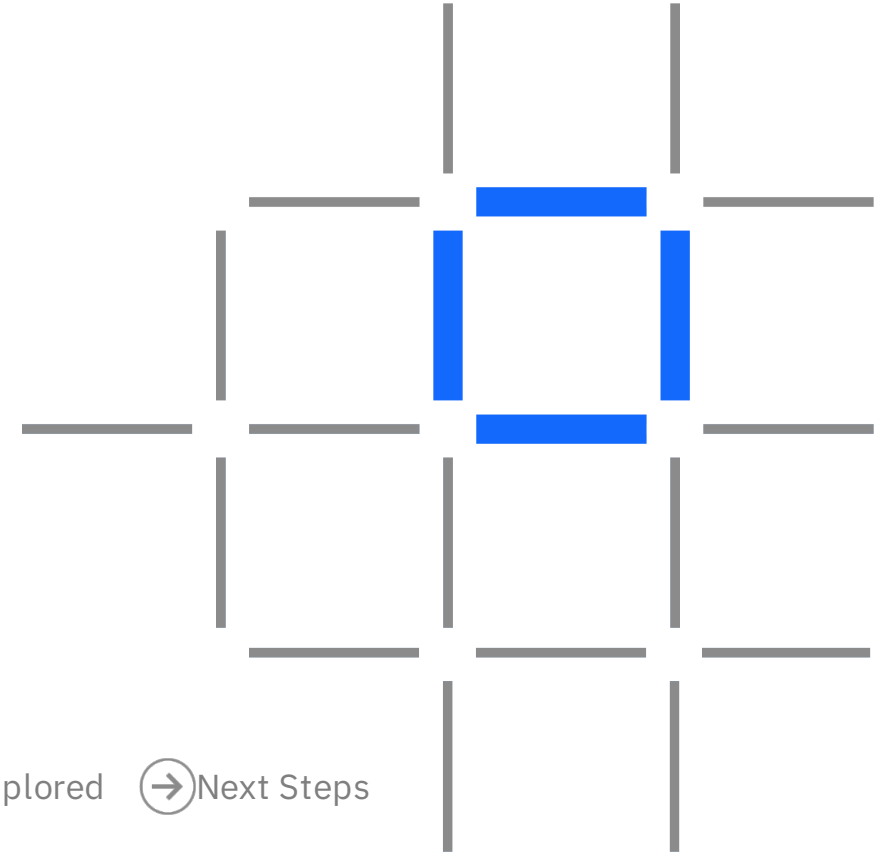
Blockchain Solutions

Use-Cases, References and How IBM Can Help

IBM **Blockchain**



Blockchain education series

Explained **Solutions** Composed Architected Explored Next Steps

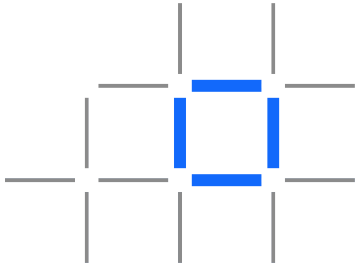
V2.02, 6 October 2017

Content

What makes a good blockchain solution?

Basics of Hyperledger Fabric & Hyperledger Composer

Your environment



Blockchain Explained

Business Network

Customers, suppliers, banks, partners, government institutions. Cross geography & regulatory boundary

Wealth is generated by the flow of goods & services across business network

Creating Markets; Public (fruit market, car auction) or Private (supply chain financing, bonds)

Digital Assets

Anything that is capable of being owned or controlled to produce value

Tangible, e.g. a diamond, car, house

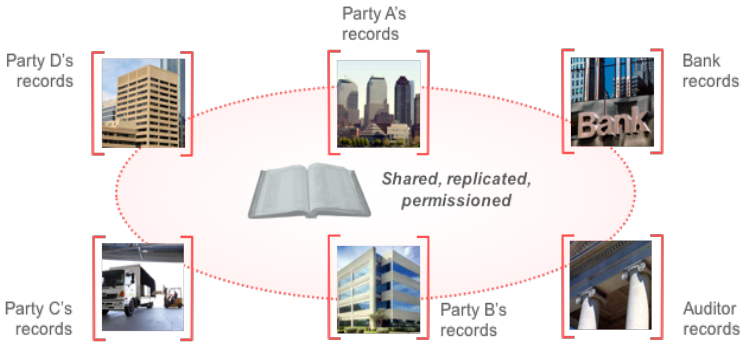
Intangible, bond [Financial], patent [**Intellectual], music [Digital]**

Ledger

Ledger is THE system of record for a institution.

Transaction – an asset transfer onto or off the ledger

Contract – conditions for transaction to occur



Blockchain

… Consensus, provenance, immutability, finality

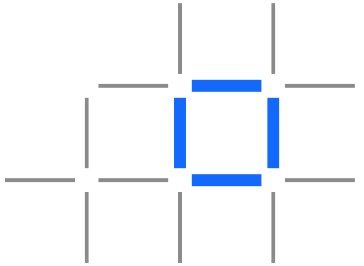
Blockchain for Business

Key Concepts:

Append-only distributed system of record shared across business network	Shared Ledger	Permissions	Ensuring appropriate visibility; transactions are secure, authenticated & verifiable
Business terms embedded in transaction database & executed with transactions	Smart Contracts	Consensus	All parties agree to network verified transaction

Benefits:

Reduces Time	Removes Cost	Reduces Risk	Enables New Business Models
Transaction time from days to near instantaneous	Overheads and cost intermediaries	Tampering, fraud & cyber crime	IoT Integration into supply chain



Good blockchain use-case or bad?

Dispute resolution

Holiday Tracking Tool

Know Your Customer

Food provenance

Track Your Child

Electronic Medical Records

What makes a good blockchain use case?

- Identifying a good blockchain use-case is not always easy!
  - However there should always be:

1. A **business problem** to be solved
   - That cannot be solved with more mature technologies
2. An identifiable **business network**
   - With Participants, Assets and Transactions
3. A need for **trust**
   - Consensus, Immutability, Finality or Provenance

What makes a good first blockchain use case?

– First use-cases are even more difficult to identify!

1. A limited scope, but still solves a real business problem
   - Minimum Viable Product in a few weeks of effort
2. A smaller business network
   - Usually without requiring regulators and consortia
3. Allows for scaling with more participants and scenarios
   - Consider shadow chains to mitigate risks

Start small, succeed and grow fast!

Ten questions to ask for the selected use case:

Understanding the business problem

1. What is the specific business problem / challenge that the first project will address?
   - Scope the business challenge up front

2. What is the current way of solving this business problem?
   - Understand current systems and areas for improvement

3. Assuming the business problem is large, what specific aspects of this business problem will be addressed by the first project?

Ten questions to ask for the selected use case:

Understanding the participants

4. Who are the business network participants (organizations) involved and what are their roles?
   - If there is no business network involved, then this is not a good use case.

5. Who are the specific people within the organization and what are their job roles?
   - Understand the key users in a business network.

Ten questions to ask for the selected use case:

Understanding the assets and transactions

6. What assets are involved and what is the key information associated with the assets?

7. What are the transactions involved, between whom, and what assets are associated with transactions?
   - Understand under what business or contractual conditions assets are under as they transfer from one owner to another.

Ten questions to ask for the selected use case:

Additional points of understanding

8. What are the main steps in the current workflow and how are these executed by the business network participants?

9. What is the expected benefit of applying blockchain technology to the business problem for each of the network participants?

10. What legacy systems are involved? What degree of integration with the legacy systems is needed?

Assessing Business Value

- It can be difficult to accurately quantify investment case for blockchain

- Things to consider:
  - Existing Pain Points
  - Scope – participants, assets, transactions
  - Benefits: baseline, minimum viable ecosystem (MVE) & mature network
  - Blockchain Design Points
  - References

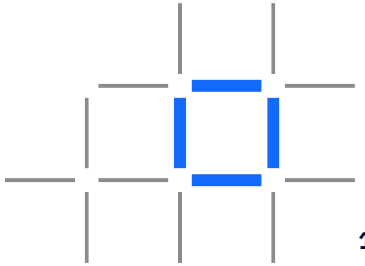
Design Thinking Workshop will help elaborate these items!

Content

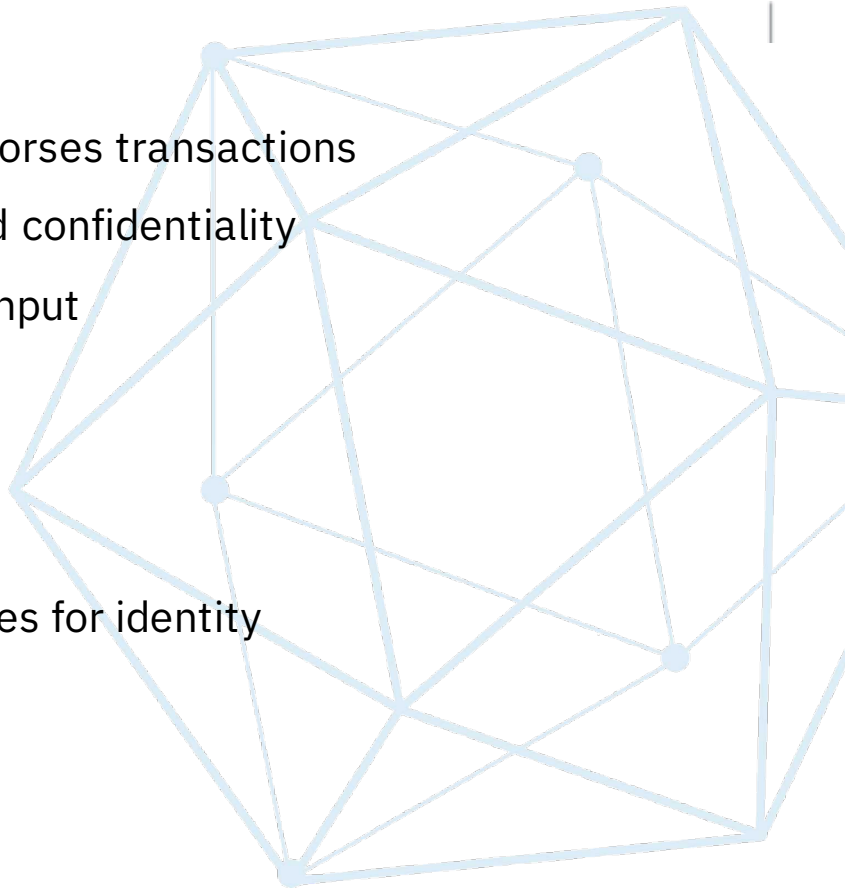
What makes a good blockchain solution?

Basics of Hyperledger Fabric & Hyperledger Composer

Your environment

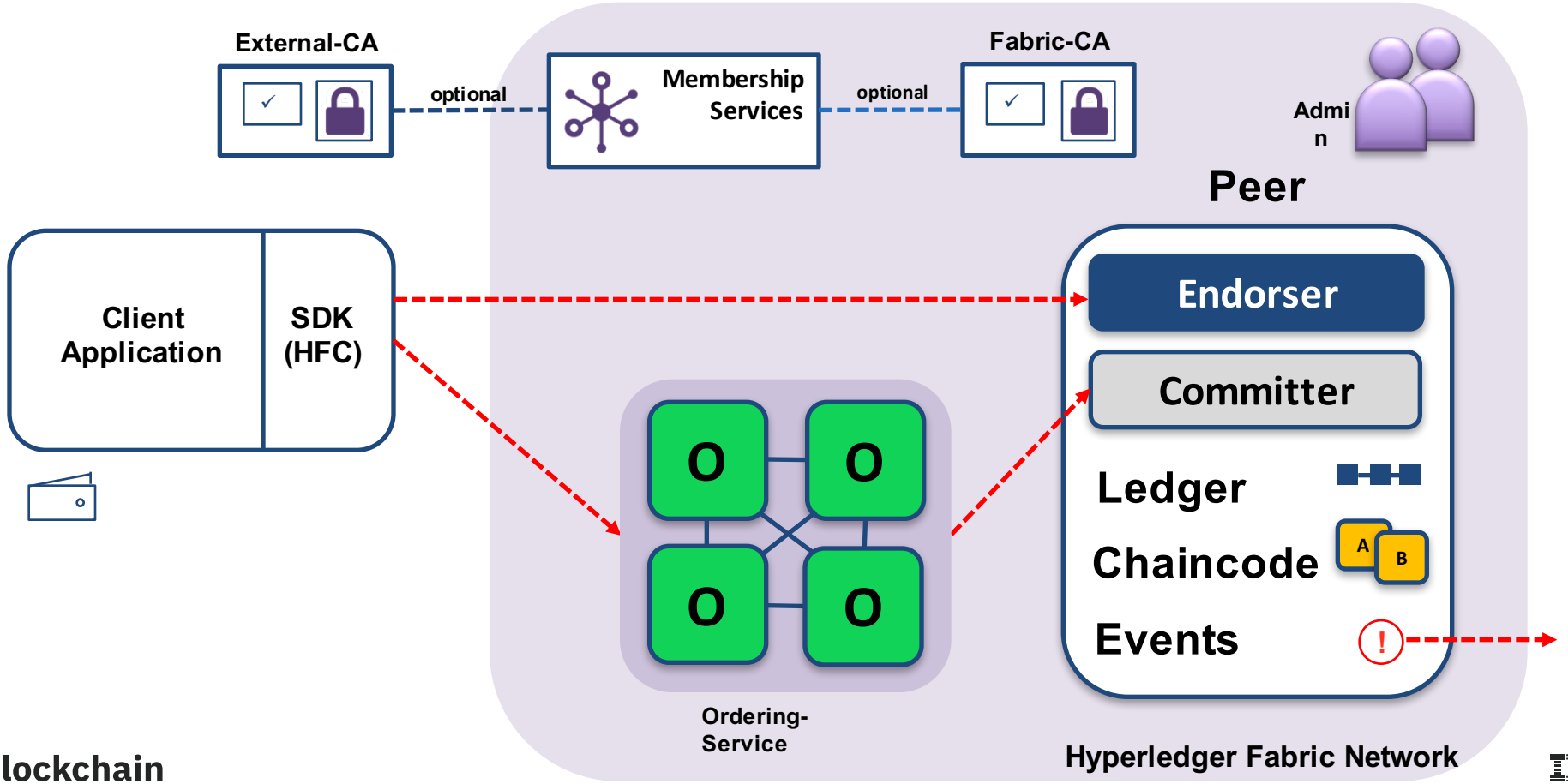


Overview of Hyperledger Fabric v1 – Design Goals

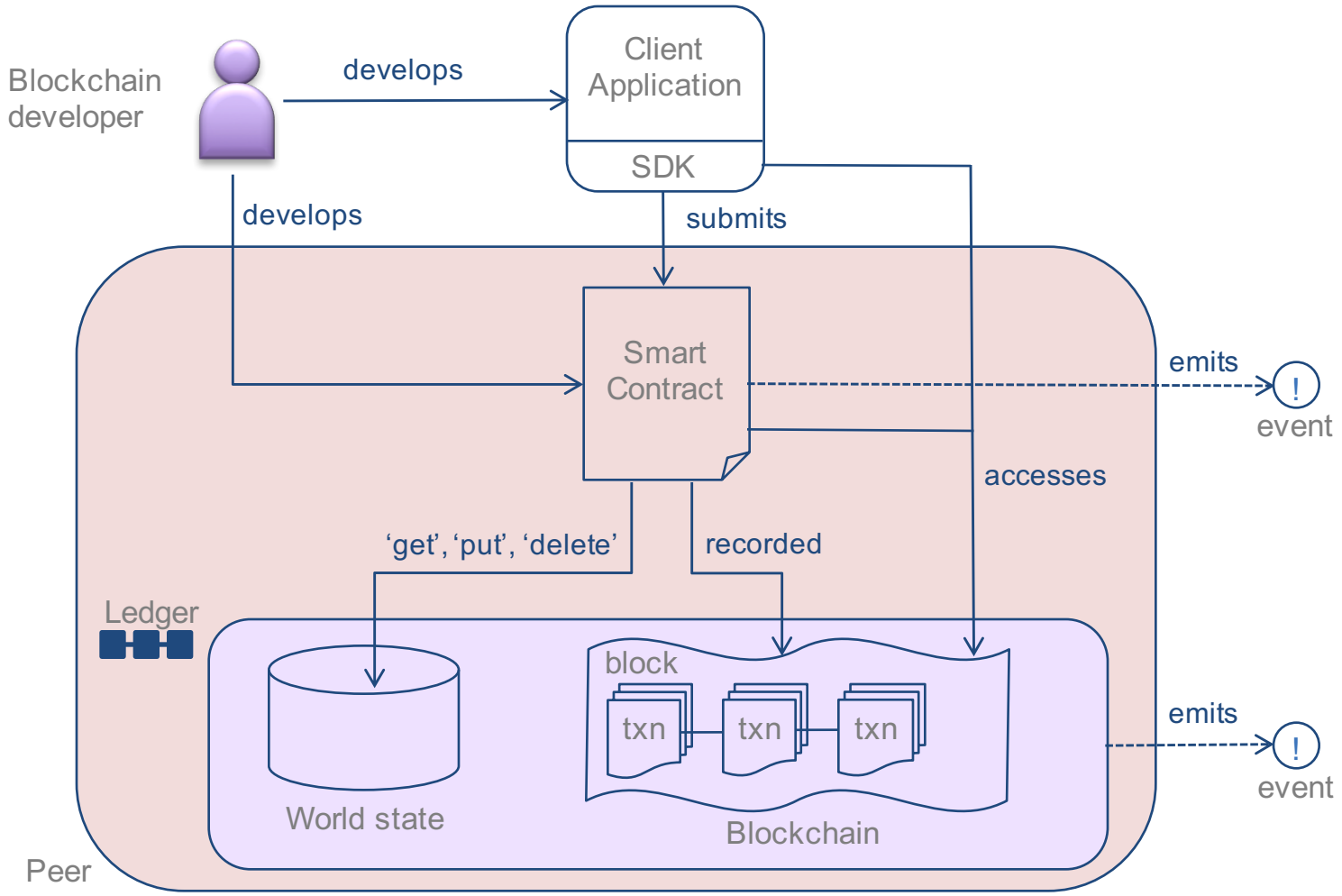


- Better reflect business processes by specifying who endorses transactions
- Support broader regulatory requirements for privacy and confidentiality
- Scale the number of participants and transaction throughput
- Eliminate non deterministic transactions
- Support rich data queries of the ledger
- Dynamically upgrade the network and chaincode
- Support for multiple credential and cryptographic services for identity
- Support for ”bring your own identity”

Hyperledger Fabric V1 Architecture

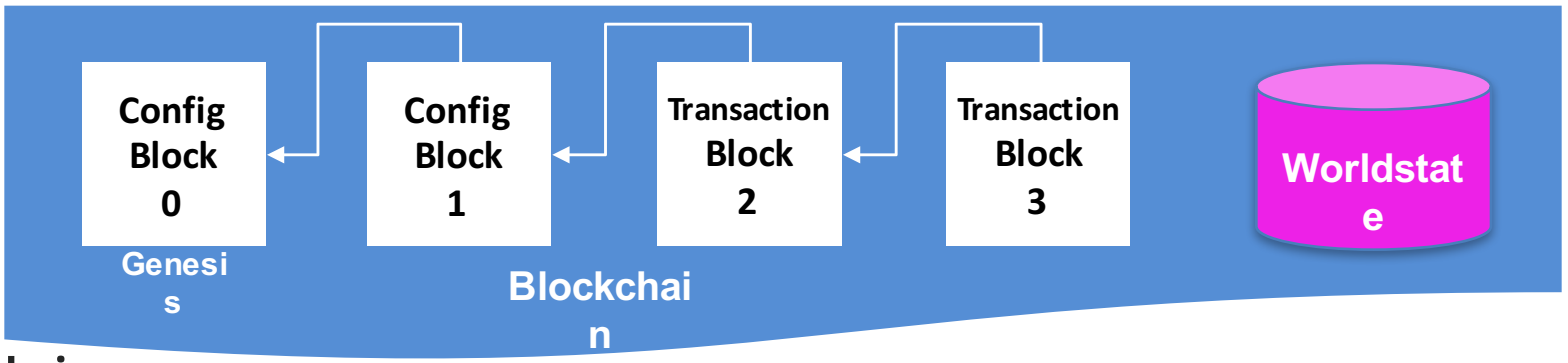


How applications interact with the ledger



Fabric Ledger

- 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
  - CouchDB is a document store that allows complex queries
- The smart contact Contract decides what is written to the worldstate

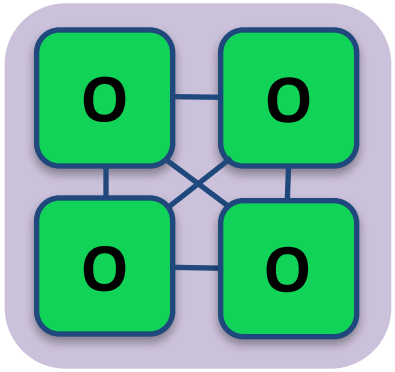


Nodes and roles

	**Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
	**Endorsing Peer:** Specialized peer also endorses transactions by receiving a transaction proposal and responds by granting or denying endorsement. Must hold smart contract.
	**Ordering Node:** 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.

Ordering Service

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

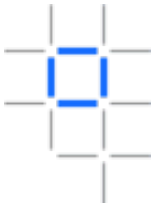


Ordering-Service

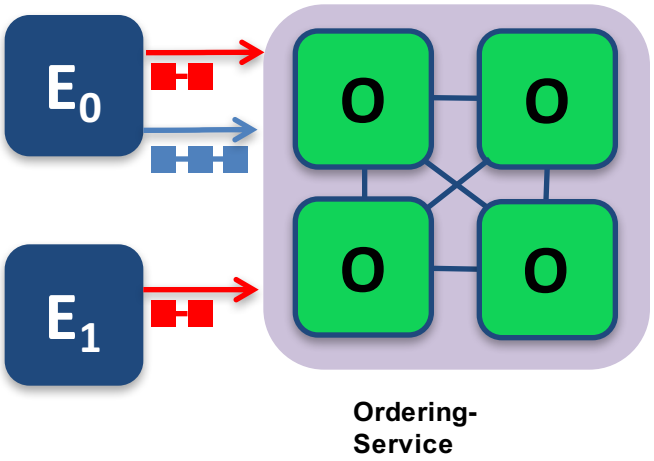
Different configuration options for the ordering service include:

- SOLO
  - Single node for development
- Kafka : Crash fault tolerant consensus
  - 3 nodes minimum
  - Odd number of nodes recommended

Channels

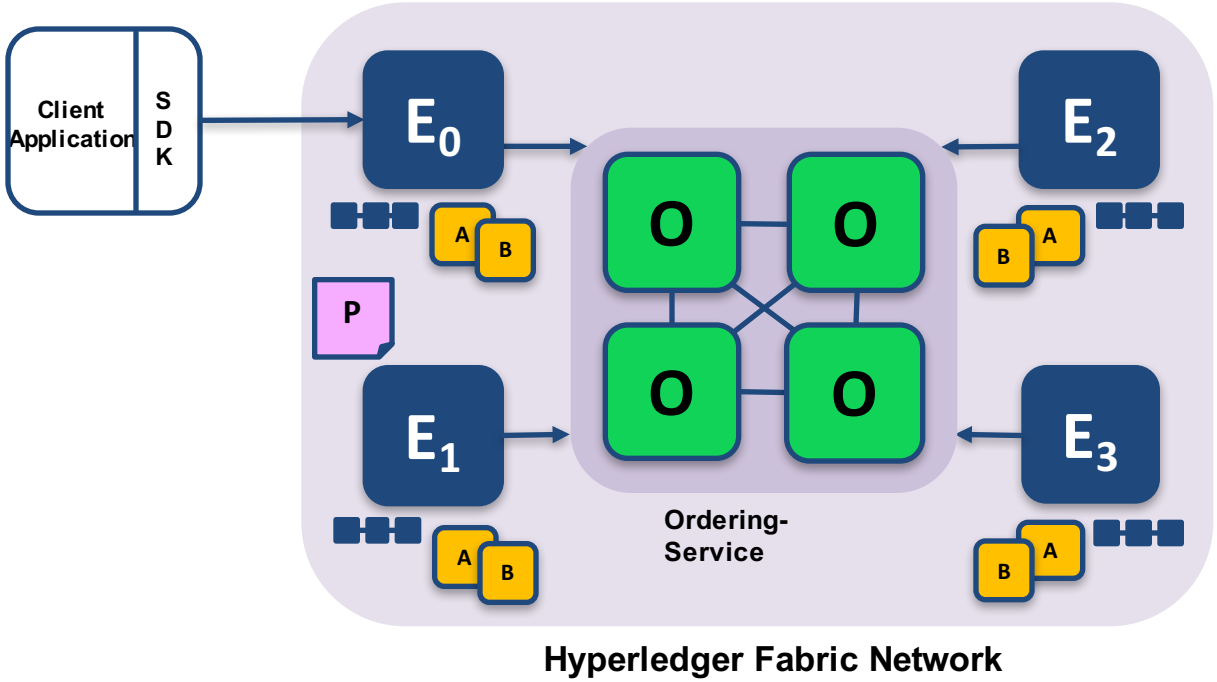


Channels provide privacy between different ledgers

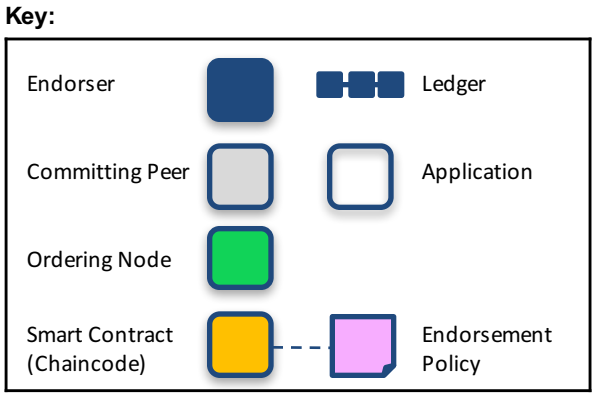


- Ledgers exist in the scope of a channel
  - Channels can be shared across an entire network of peers
  - Channels can be permissioned for a specific set of participants
- Chaincode is installed on peers to access the worldstate
- Chaincode is instantiated on specific channels
- Peers can participate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

Single Channel Network



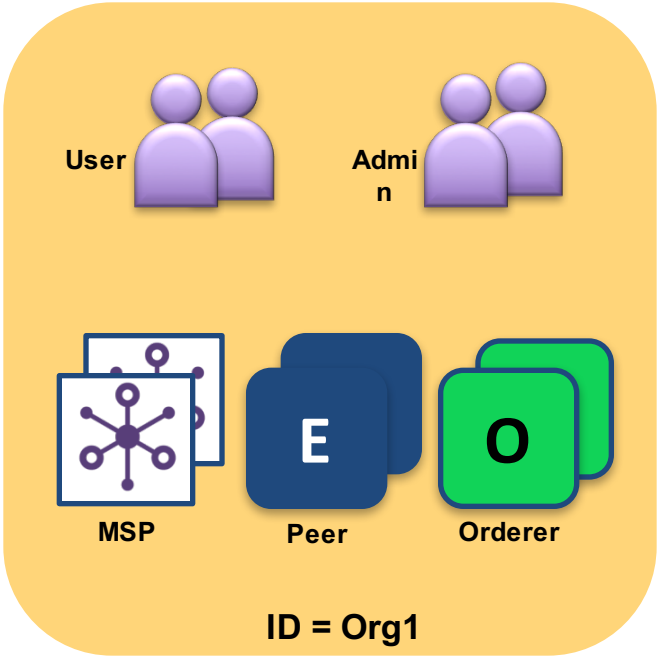
- Similar to v0.6 PBFT model
- All peers connect to the same system channel (blue).
- All peers have the same chaincode and maintain the same ledger
- Endorsement by peers E_0, E_1, E_2 and E_3



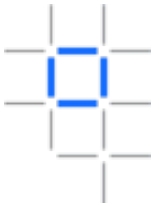
Organisations

Organisations define boundaries within a Fabric Blockchain Network

- Each organisation defines:
  - Membership Services Provider (MSP) for identities
  - Administrator(s)
  - Users
  - Peers
  - Orderers (optional)
- A network can include many organisations representing a consortium
- Each organisation has an ID

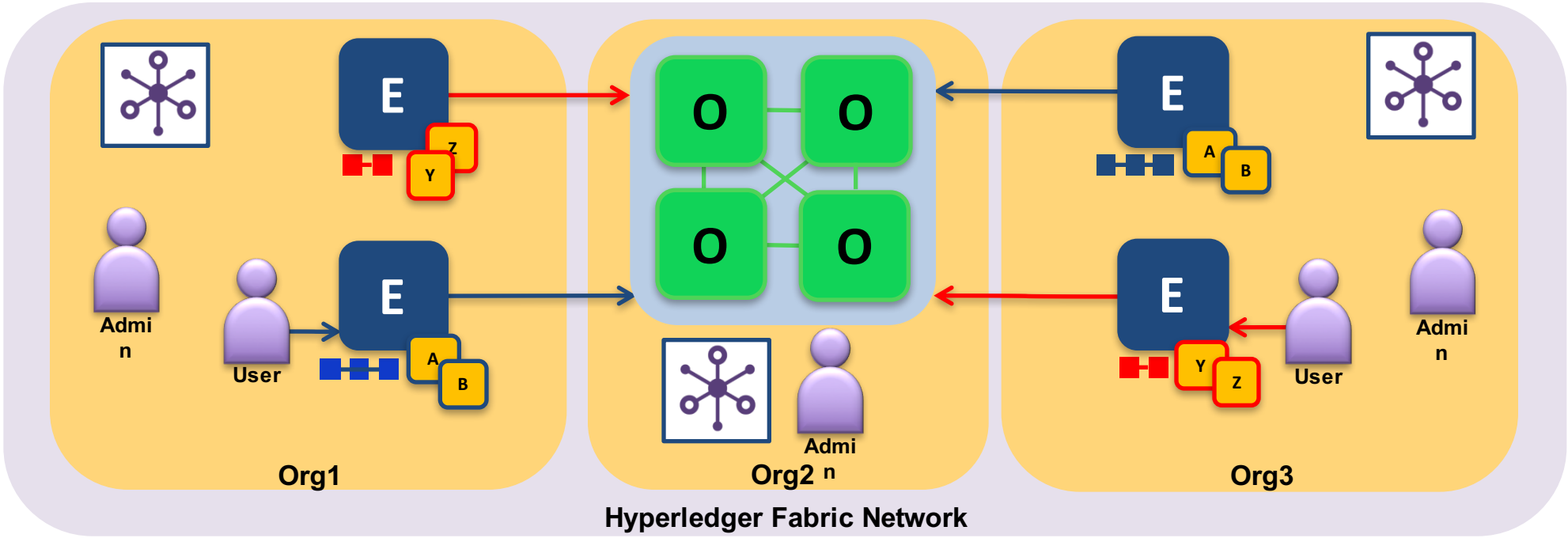


Consortium Network

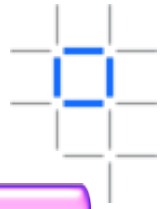


An example consortium network of 3 organisations

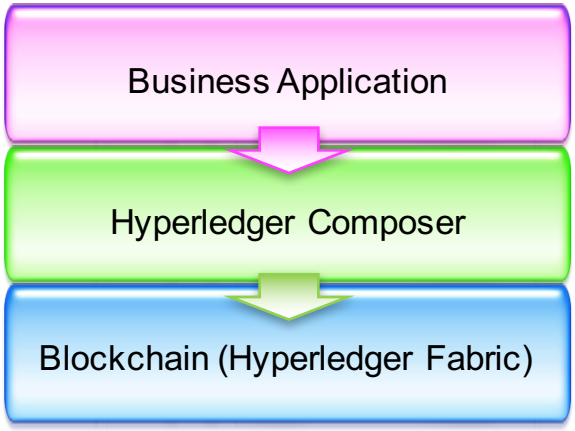
- Orgs 1 and 3 run peers
- Org 2 provides the ordering service only



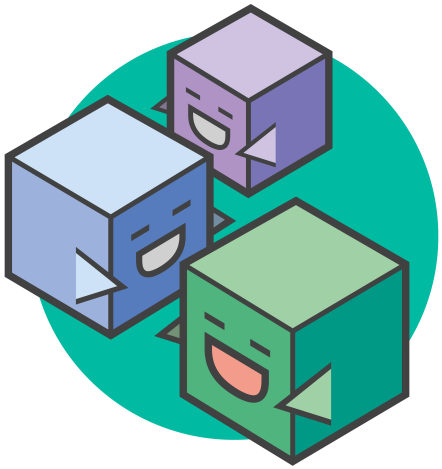
Hyperledger Composer: Accelerating time to value



https://hyperledger.github.io/composer/



- A suite of high level application abstractions for business networks
- Emphasis on business-centric vocabulary for quick solution creation

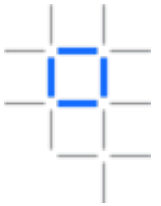


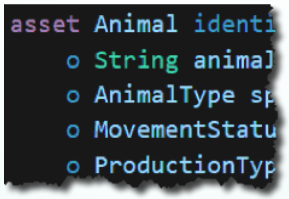
crease understanding and flexibility

- Features
  - Model your business networks, test and expose via APIs
  - Applications invoke APIs transactions to interact with business network
  - Integrate existing systems of record using loopback/REST
- Fully open and part of Linux Foundation Hyperledger
- Try it in your web browser now: http://composer-playground.mybluemix.net/

IBM **Blockchain**

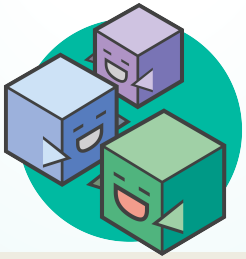
Extensive, Familiar, Open Development Toolset





Data modelling

JavaScript business logic



Web playground

Client libraries

Editor support

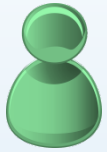
CLI utilities

Code generation

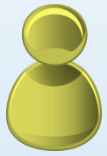
Existing systems and data

User Roles in a Blockchain Project

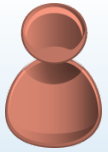
- **Network Service Provider**
  - Governs the network: channels, membership etc.
  - A consortium of network members or designated authority



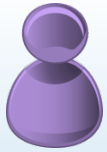
- **Network Service Consumer**
  - Operates a set of peers and certificate authorities on the network
  - Represents an organization on the business network



- **Business Service Provider**
  - Develops blockchain business applications
  - Includes transaction, app server, integration and presentation logic



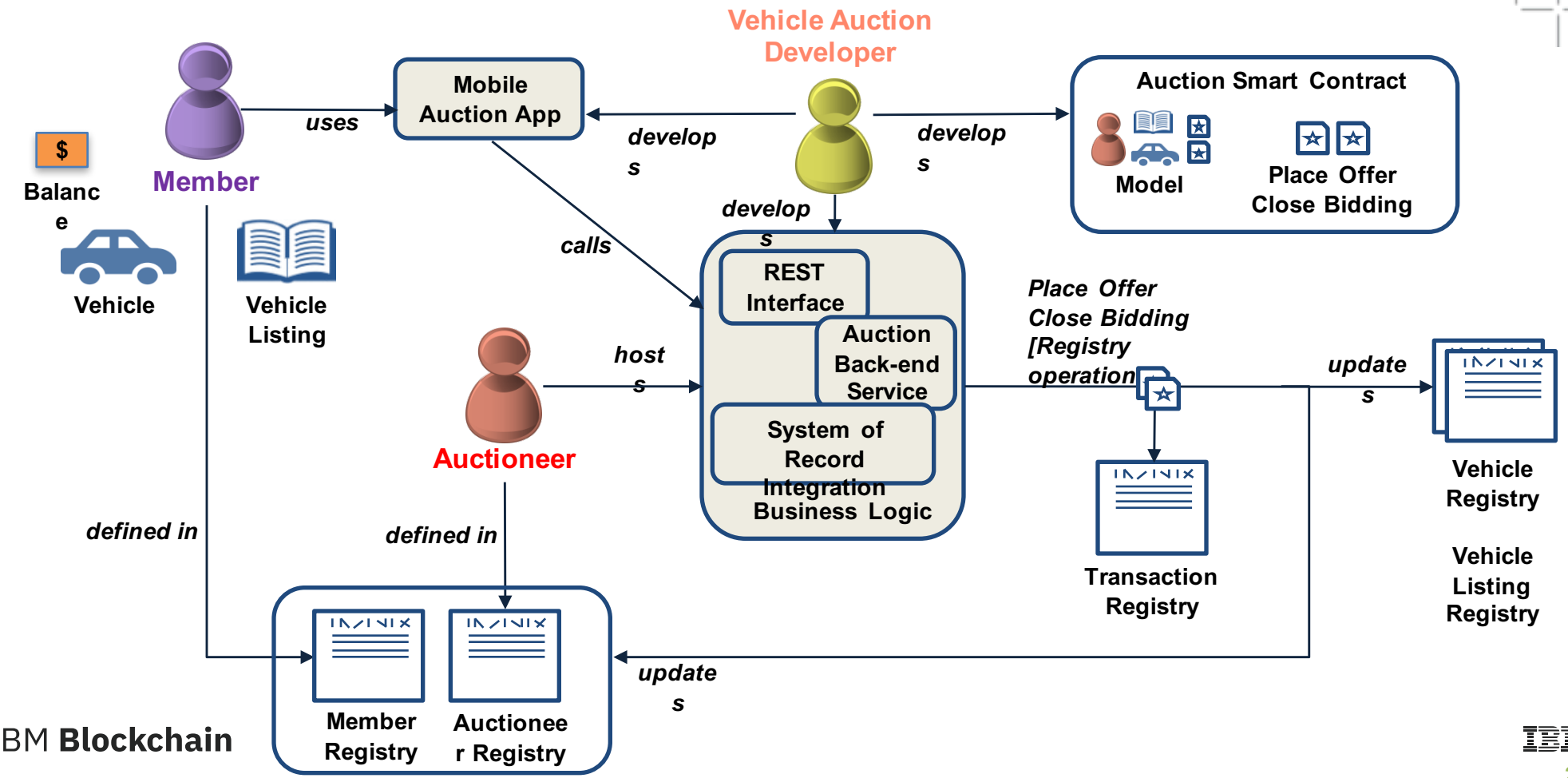
- **Business Service Consumer**
  - Hosts application and integration logic which invokes blockchain transactions



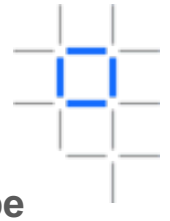
- **End-user**
  - Runs presentation logic e.g. on mobile device or dashboard

A single organization may play multiple rol

Key Concepts for a Vehicle Auction Developer



Access Control



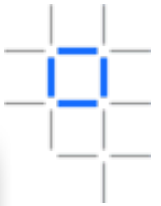
```
rule EverybodyCanReadEverything {
    description: "Allow all participants read access to all resources"
    participant: "org.acme.sample.SampleParticipant"
    operation: READ
    resource: "org.acme.sample.*"
    action: ALLOW
}
```

```
rule OwnerHasFullAccessToTheirAssets {
    description: "Allow all participants full access to their assets"
    participant(p): "org.acme.sample.SampleParticipant"
    operation: ALL
    resource(r): "org.acme.sample.SampleAsset"
    condition: (r.owner.getIdentifier() === p.getIdentifier())
    action: ALLOW
}
```

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

- It is possible to restrict which resources can be read and modified by which participants
  - Rules are defined in an .acl file and deployed with the rest of the model
  - Transaction processors can also look up the current user and implement rules programmatically
- ACL rules can be simple (e.g. everybody can read all resources) or more complex (e.g. only the owner of an asset can do everything to it)
- Application supplies credentials (userid/secret) of the participant when connecting to the Fabric network
  - This also applies to Playground!
  - Remember to grant System ACL all access if necessary

Events and Queries



- Events allow applications to take action when a transaction occurs
  - Events are defined in models
  - Events are emitted by transaction processor scripts
  - Events are caught by business applications
- Caught events include transaction ID and other relevant information

```
event SampleEvent {
  --> SampleAsset asset
  o String oldValue
  o String newValue
}
```

```
// Emit an event for the modified asset.
var event = getFactory().newEvent('org.acme.sample', 'SampleEvent');
event.asset = tx.asset;
event.oldValue = oldValue;
event.newValue = tx.newValue;
emit(event);
```

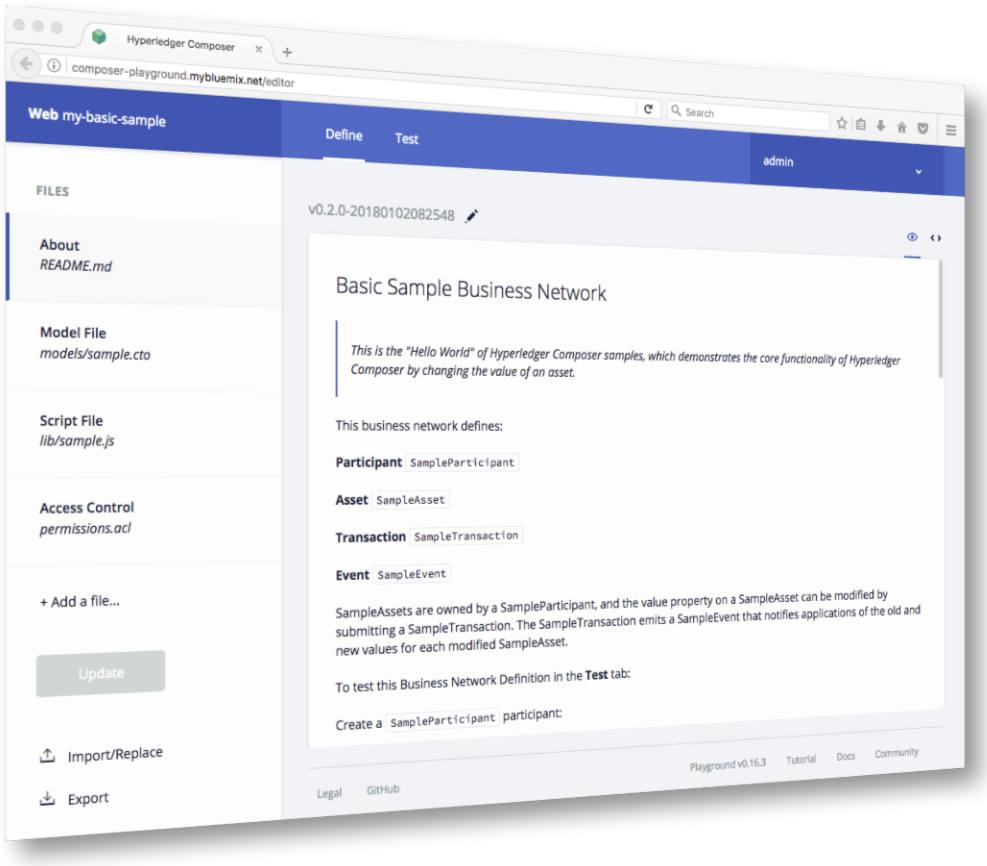
```
businessNetworkConnection.on('SampleEvent', (event) => {
    console.log(event);
}
```

- Queries allow applications to perform complex registry searches
  - They can be statically defined in a separate .qry file or generated dynamically by the application
  - They are invoked in the application using *buildQuery*() or *query*()
  - Queries require the blockchain to be backed by CouchDB

```
query selectCommoditiesWithHighQuantity {
  description: "Select commodities based on quantity"
  statement:
      SELECT org.acme.trading.Commodity
          WHERE (quantity > 60)
}
```

```
return query('selectCommoditiesWithHighQuantity', {})
```

Smart Contract Development: Composer Playground



- Web tool for defining and testing Hyperledger Composer models and scripts
- Designed for the application developer
  - Define assets, participants and transactions
  - Implement transaction processor scripts
  - Test by populating registries and invoking transactions
- Deploy to instances of Hyperledger Fabric V1, or simulate completely within browser
- Install on your machine or run online at http://composer-playground.mybluemix.net

General purpose development: Visual Studio Code

- Composer extension available for this popular tool

- Features to aid rapid Composer development
  - Edit all Composer file types with full syntax highlighting
  - Validation support for models, queries and ACLs
  - Inline error reporting
  - Snippets (press Ctrl+Space for code suggestions)
  - Generate UML diagrams from models

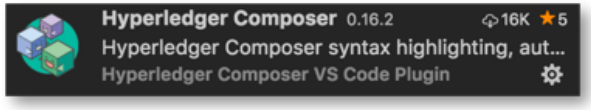
- Install directly from Code Marketplace

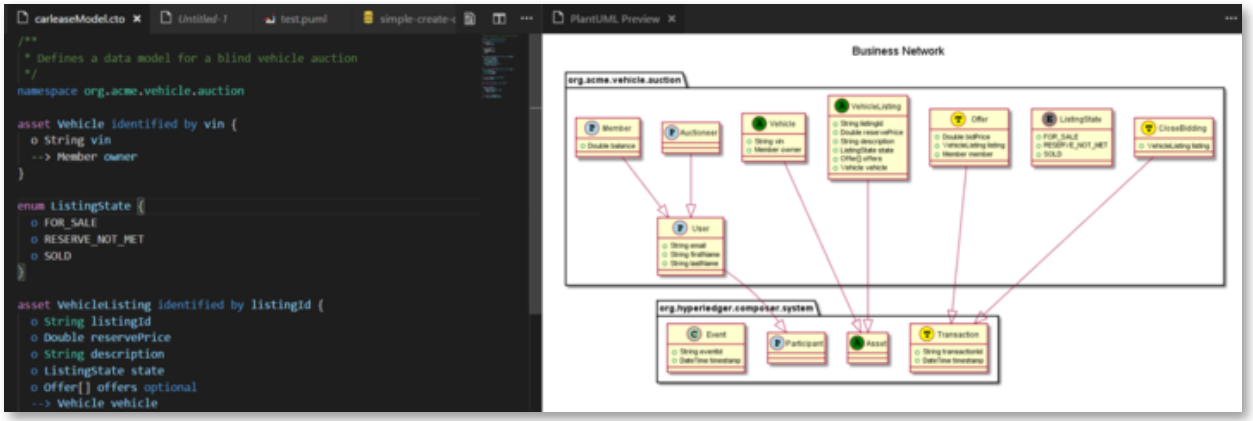
```
namespace org.acme.vehicle.lifecycle

import composer.base.Person
import composer.business.Business

participant PrivateOwner identified by email extends Person {
  o String email
}
```

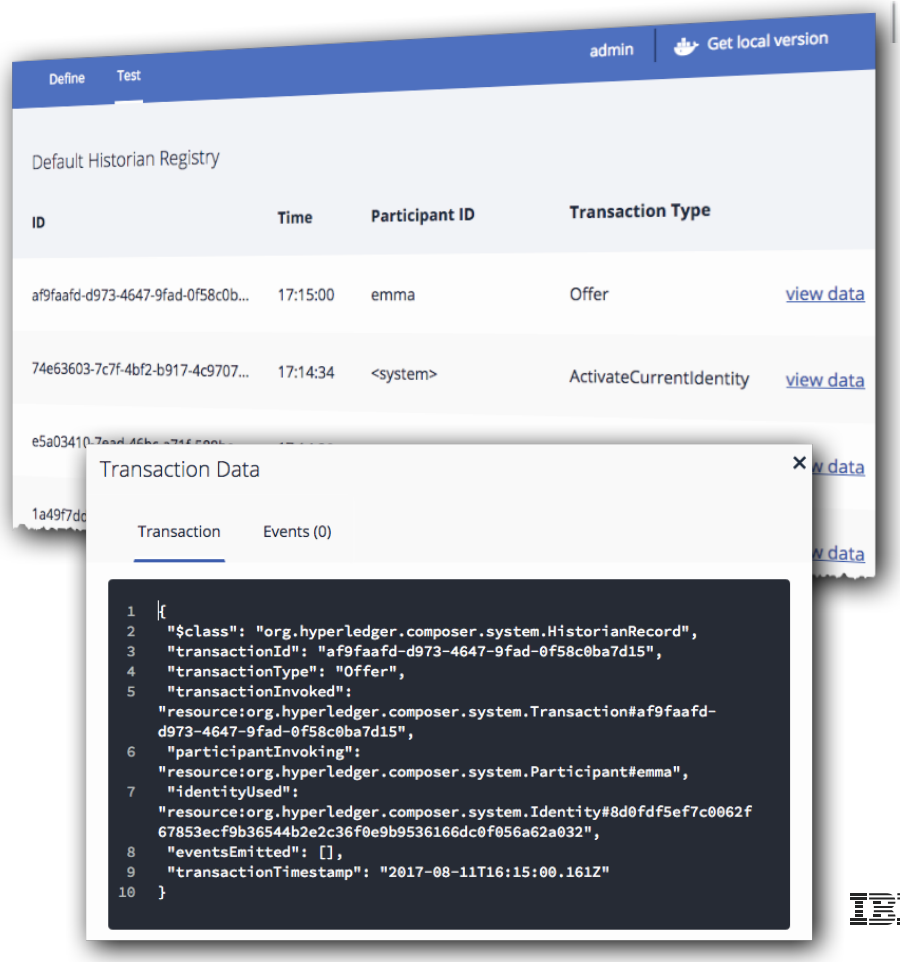
```
[Composer] IllegalModelException: Could not find super type Pea
rson
participant PrivateOwner identified by email extends Pearson {
  o String email
}
```



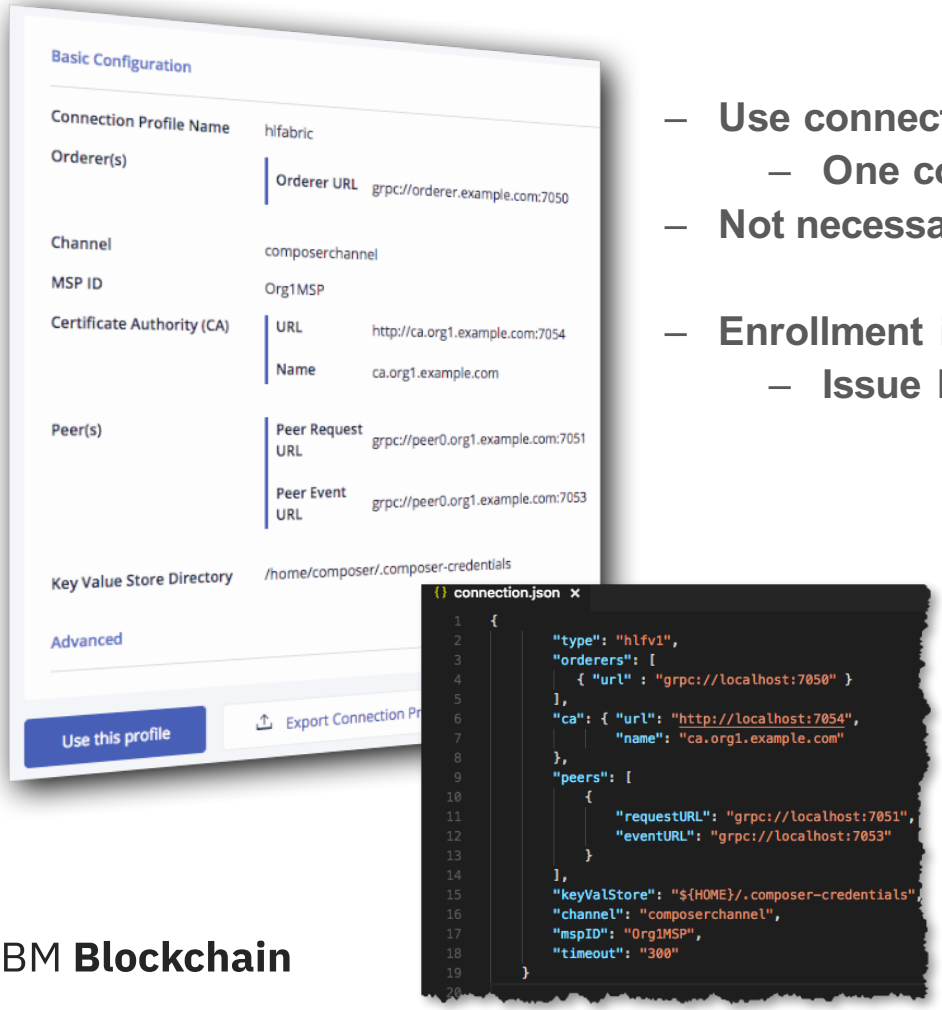


Debugging

- Playground Historian allows you to view all transactions
  - See what occurred and when
- Diagnostics framework allows for application level trace
  - Uses the *Winston* Node.js logging framework
  - Application logging using DEBUG env var
  - Composer Logs sent to stdout and ./logs/trace_<processid>.trc
- Fabric chaincode tracing also possible (see later)
- More information online: https://hyperledger.github.io/composer/problems/diagnostics.html



Connection Profiles to Hyperledger Fabric



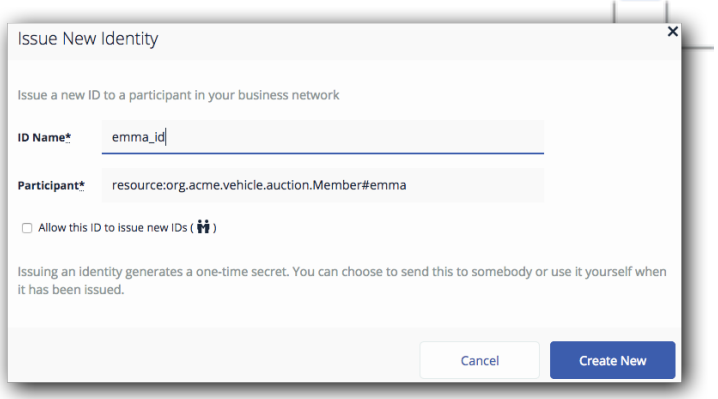
- Use connection profiles to describe Fabric connection parameters
  - One connection profile required per channel
- Not necessary for web-based simulation

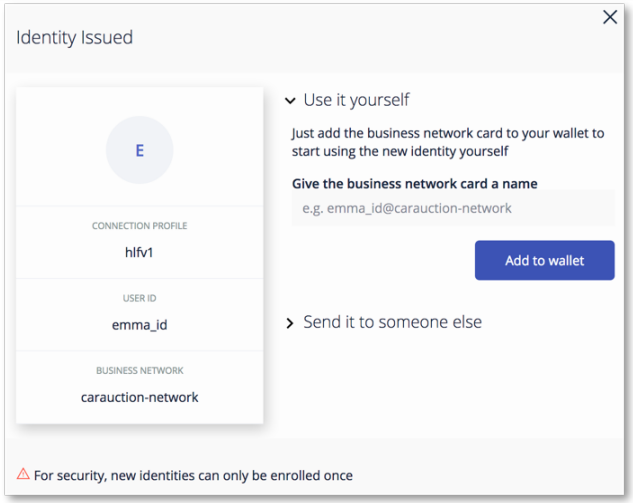
- Enrollment in Hyperledger Fabric network required (see later)
  - Issue Fabric identity from Composer participants

- Connection profiles currently used by Composer only
  - Plans to implement common connection profiles that can be used by both Fabric and Composer

Participant Identity

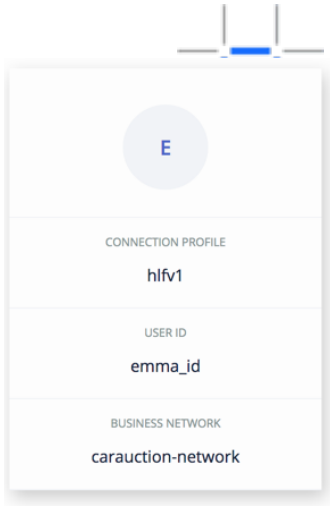
- The **Network Service Consumer** issues network participants with an identity in order to connect to Hyperledger Fabric
  - Issued as a Hyperledger Fabric userid/secret
  - Automatically swapped for a certificate on first use
  - Packaged in a Business Network Card and supplied when the client application connects
- Composer Participant to Fabric Identity mapping is stored on the blockchain in an *identity registry*
- Usually, only **Business Service Consumers** have a Fabric identity
  - **End-users** log in to the business application using a separately managed identity; blockchain transactions invoked by proxy
- Manage identity from Playground, Javascript, REST or command line
  - For example: Test connection, issue identity, bind an identity to a participant, revoke an identity, list identities

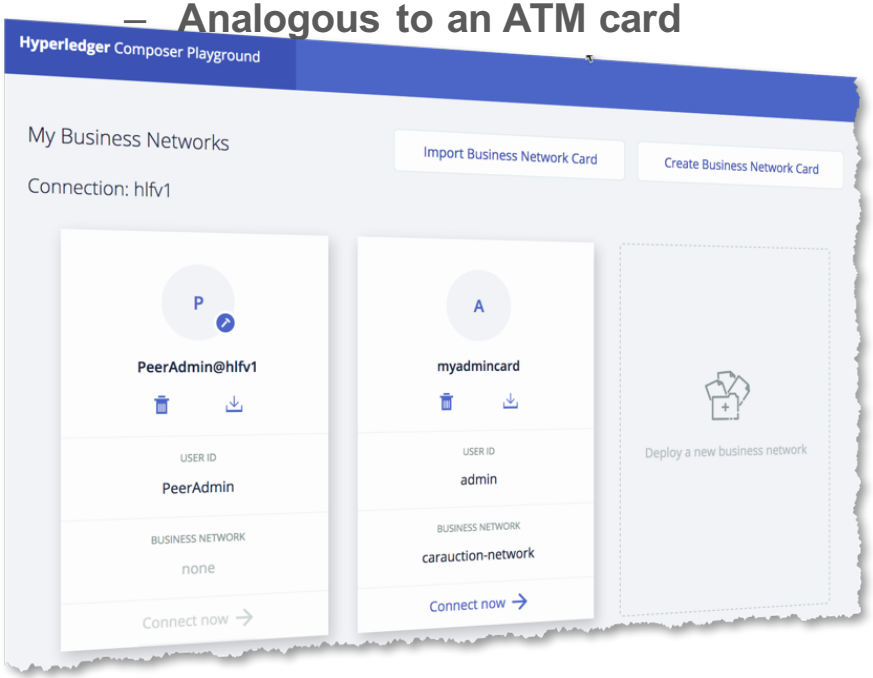




Business Network Cards

- Business Network Cards are a convenient packaging of *identity* and *connection profile*
  - Contains everything you need to connect to blockchain business network
  - Each card refers to a single participant and single business network
  - Analogous to an ATM card





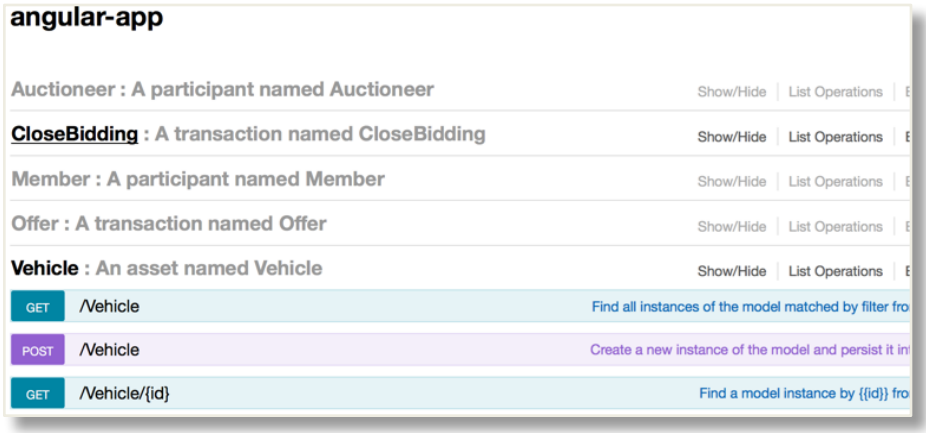
- Manage cards from both Playground and command-line
  - Create, delete, export, import, list
  - Create requires userid/secret or certificate/private key
- Use cards to connect to Fabric from Playground, application

```
composer network install -a my.bna -c my.card
```

```
// Connect and log in to HLF
var businessNetwork = new BusinessNetworkConnection();
return businessNetwork.connect('cardName')
.then(function(businessNetworkDefinition){
    // Connected
});
```

Systems of Record Integration

- Domain specific APIs very attractive to mobile and web developers. Resources and operations are business-meaningful
- Composer exploits Loopback framework to create REST APIs: https://loopback.io/
- Extensive test facilities for REST methods using loopback
- Secured using JS Passport, giving >400 options for authentication
- Composer provides back-end integration with any loopback compatible product
  - e.g. IBM Integration Bus, API Connect, StrongLoop
  - Outbound and Inbound (where supported by middleware)

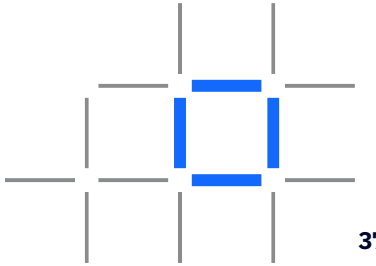


Content

What makes a good blockchain solution?

Basics of Hyperledger Fabric & Hyperledger Composer

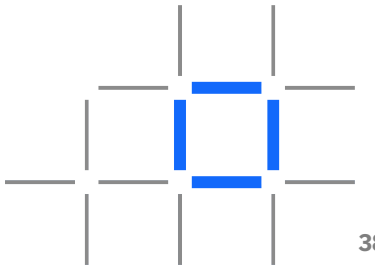
Your environment



High level steps

1. Get access to LinuxONE Community Cloud
2. Start up your SLES guest
3. Run a script to setup your environment
4. Verify environment setup
5. (Optional) Run development lab to work through coding

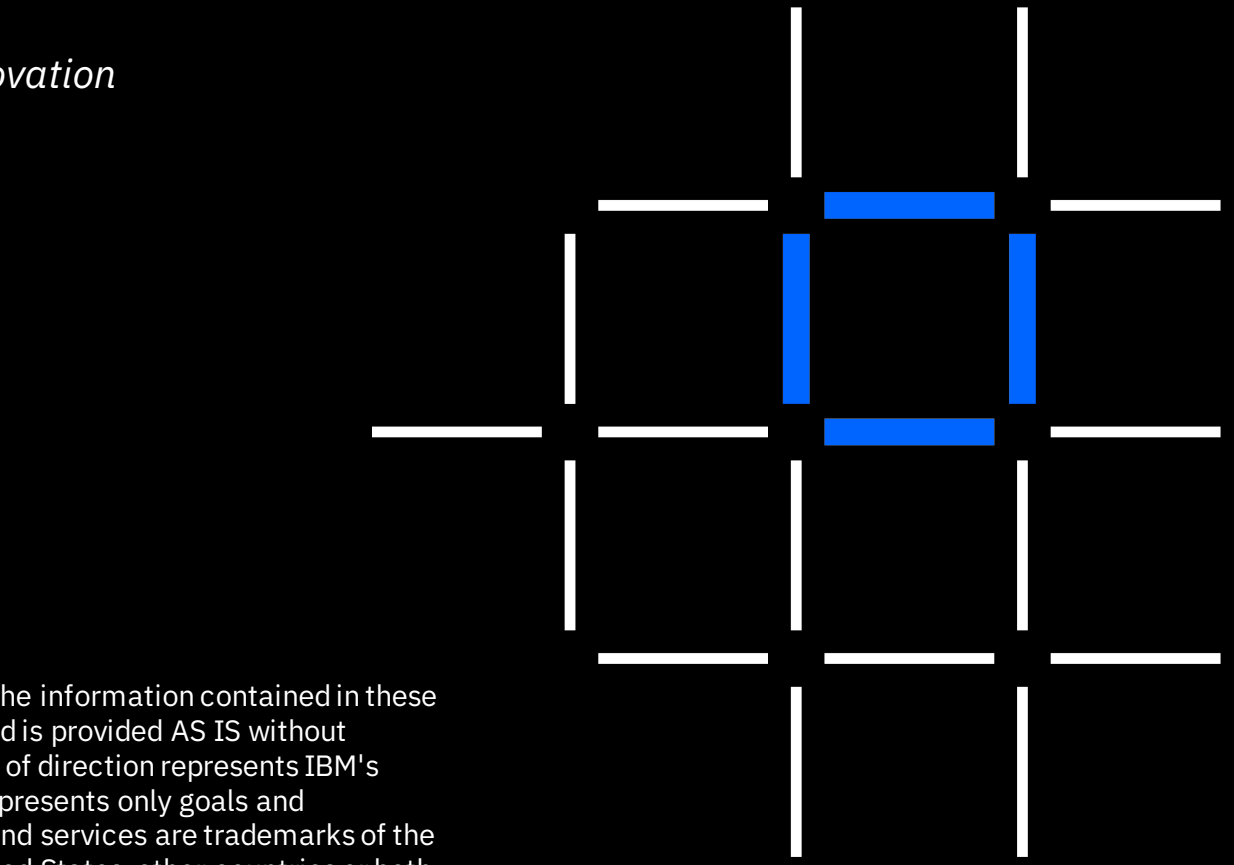
https://github.com/zsteelman/UofA



Thank you

Jennifer Foley
Integration Architect
Worldwide Client Center for Systems Innovation
foleyje@us.ibm.com

IBM **Blockchain**

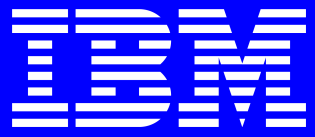


www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org

IBM


IBM