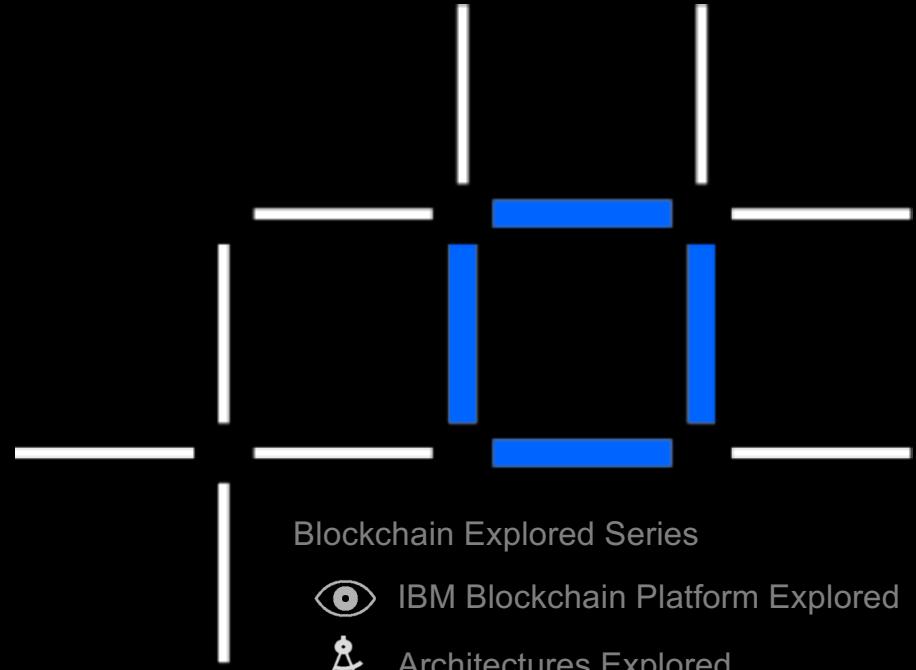


Composer Explored

A Technical Introduction to Hyperledger Composer



Blockchain Explored Series

-  IBM Blockchain Platform Explored
-  Architectures Explored
-  Fabric Explored
-  **Composer Explored**
-  What's New

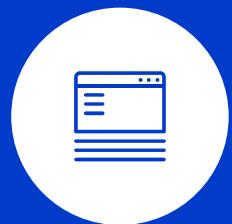
V2.10, 23 May 2018

IBM Blockchain





What is Hyperledger Composer?



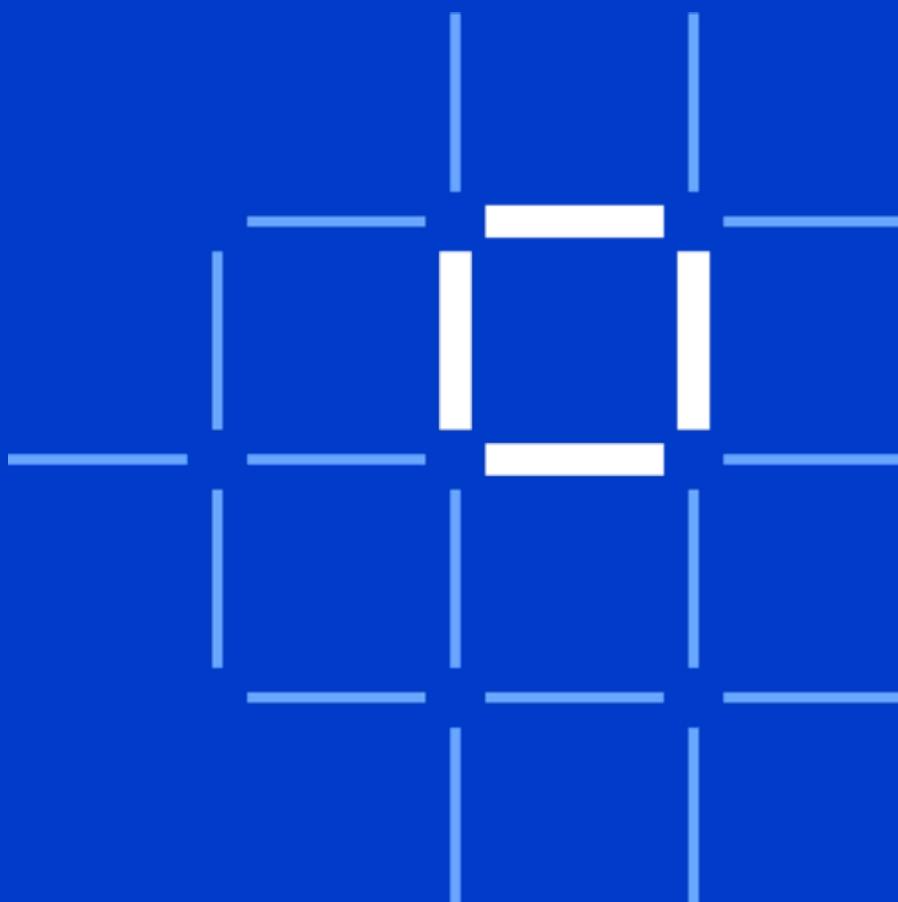
Application Development

*Writing the application
Modeling the business network*

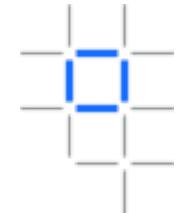


Effective Administration

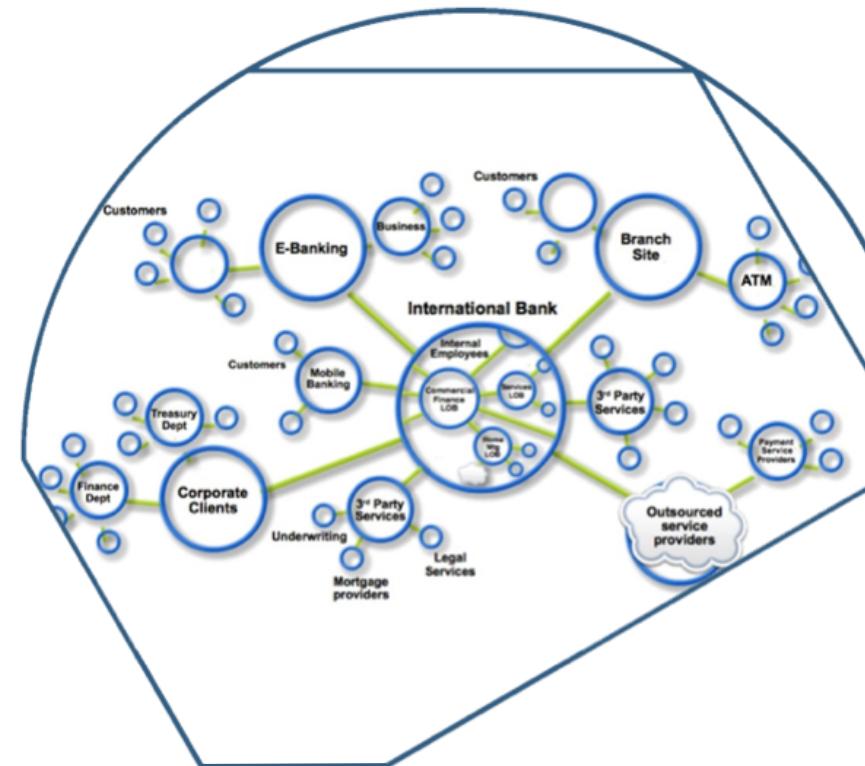
*Deploying to a blockchain
Interacting with systems of record*



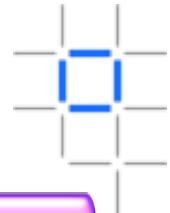
Blockchain Recap



- Blockchain builds on basic business concepts
 - **Business Networks** connect businesses
 - **Participants** with Identity
 - **Assets** flow over business networks
 - **Transactions** describe asset exchange
 - **Contracts** underpin transactions
 - The **ledger** is a log of transactions
- Blockchain is a shared, replicated ledger
 - Consensus, immutability, finality, provenance

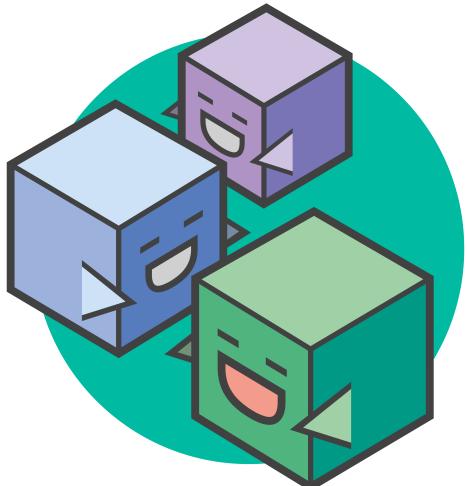


Hyperledger Composer: Accelerating time to value

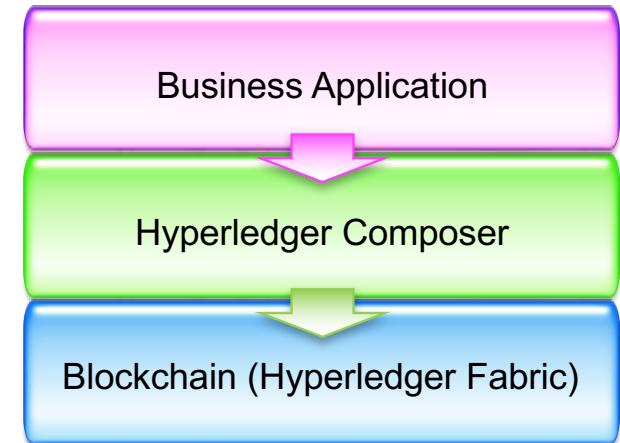


<https://hyperledger.github.io/composer/>

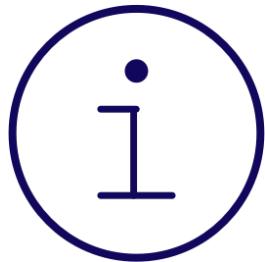
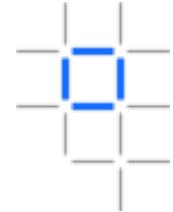
- A suite of high level application **abstractions** for business networks
- Emphasis on business-centric vocabulary for quick solution creation
- Reduce risk, and increase understanding and flexibility



- 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/>

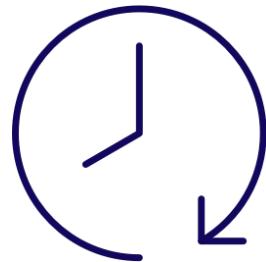


Benefits of Hyperledger Composer



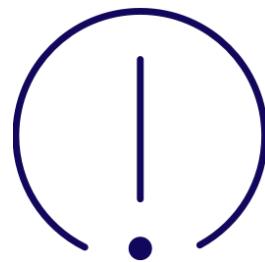
**Increases
understanding**

Bridges simply from
business concepts to
blockchain



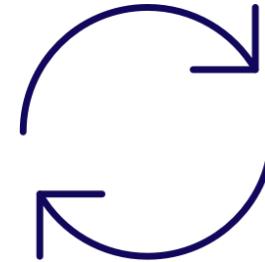
**Saves
time**

Develop blockchain
applications more
quickly and cheaply



**Reduces
risk**

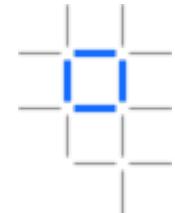
Well tested, efficient
design conforms to
best practice



**Increases
flexibility**

Higher level
abstraction makes it
easier to iterate

Extensive, Familiar, Open Development Toolset

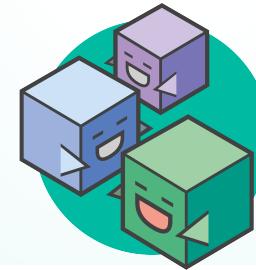


```
asset Animal identi  
  o String anima]  
  o AnimalType sp  
  o MovementStatu  
  o ProductionTyp
```

Data modelling



JavaScript
business logic



Web playground

composer-client
composer-admin



Client libraries



Editor support

\$ composer

CLI utilities



Code generation

Powered by
 LoopBack
Node.js Framework

Swagger

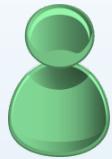
Existing systems and
data

User Roles in a Blockchain Project



- Network Service Provider

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



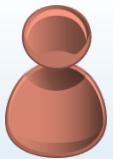
- Network Service Consumer

- **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 roles!

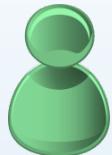
The Developer Role in a Blockchain Project



- 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

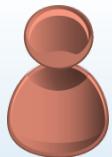
- Represents an organization on the business network



- 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



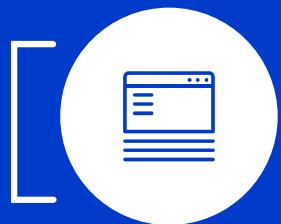
- End-user

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

A single organization may play multiple roles!



What is Hyperledger Composer?



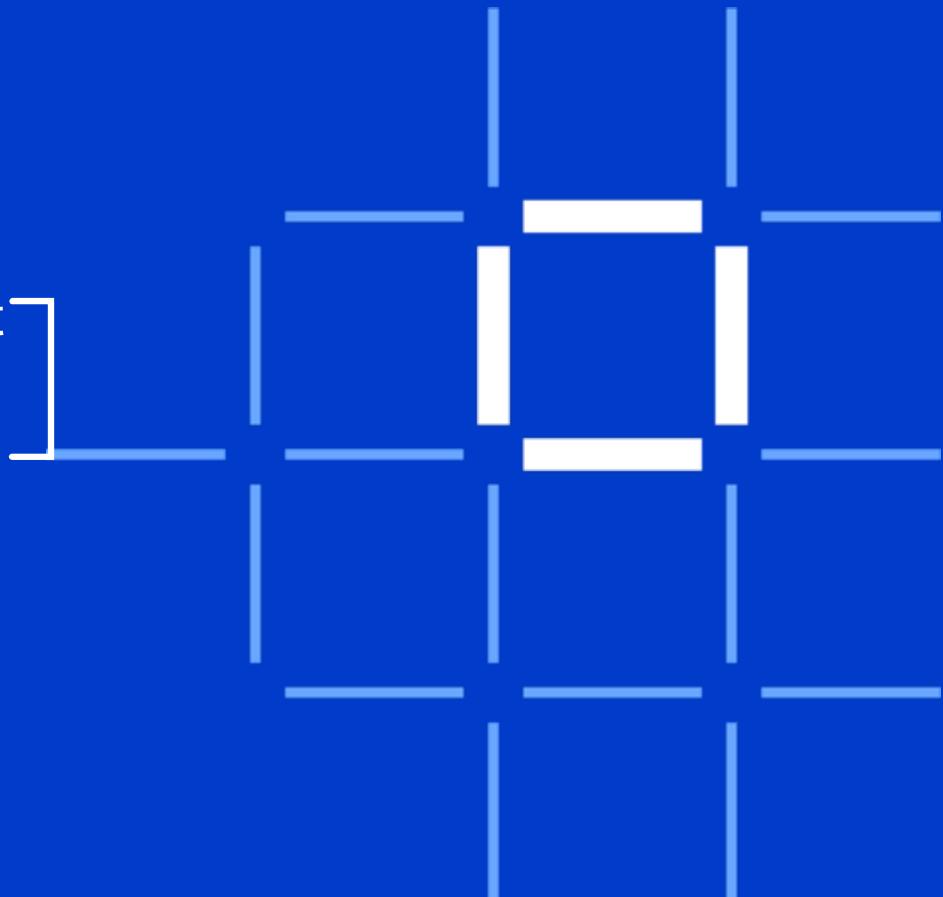
Application Development

*Writing the application
Modeling the business network*

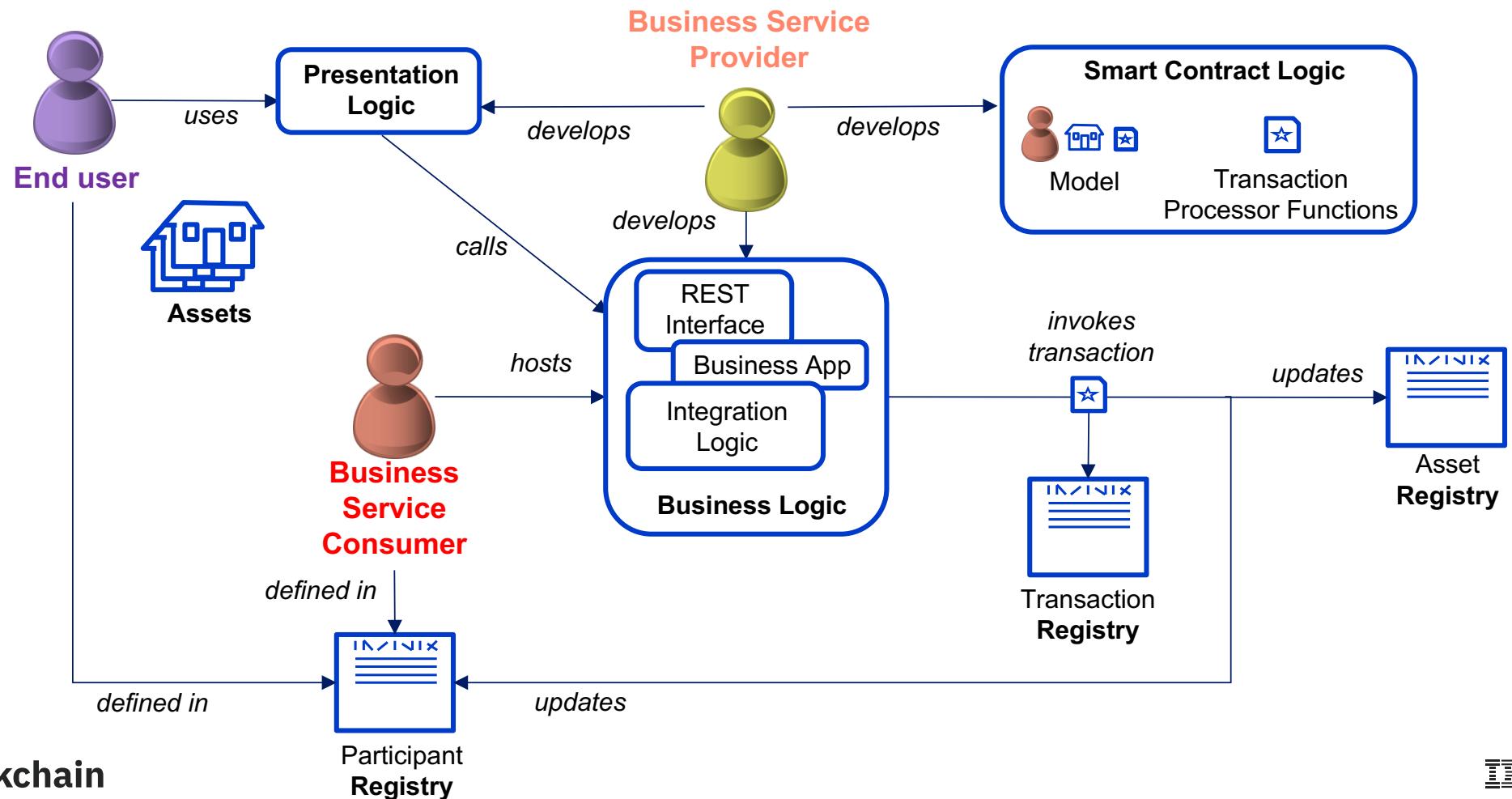
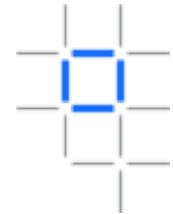


Effective Administration

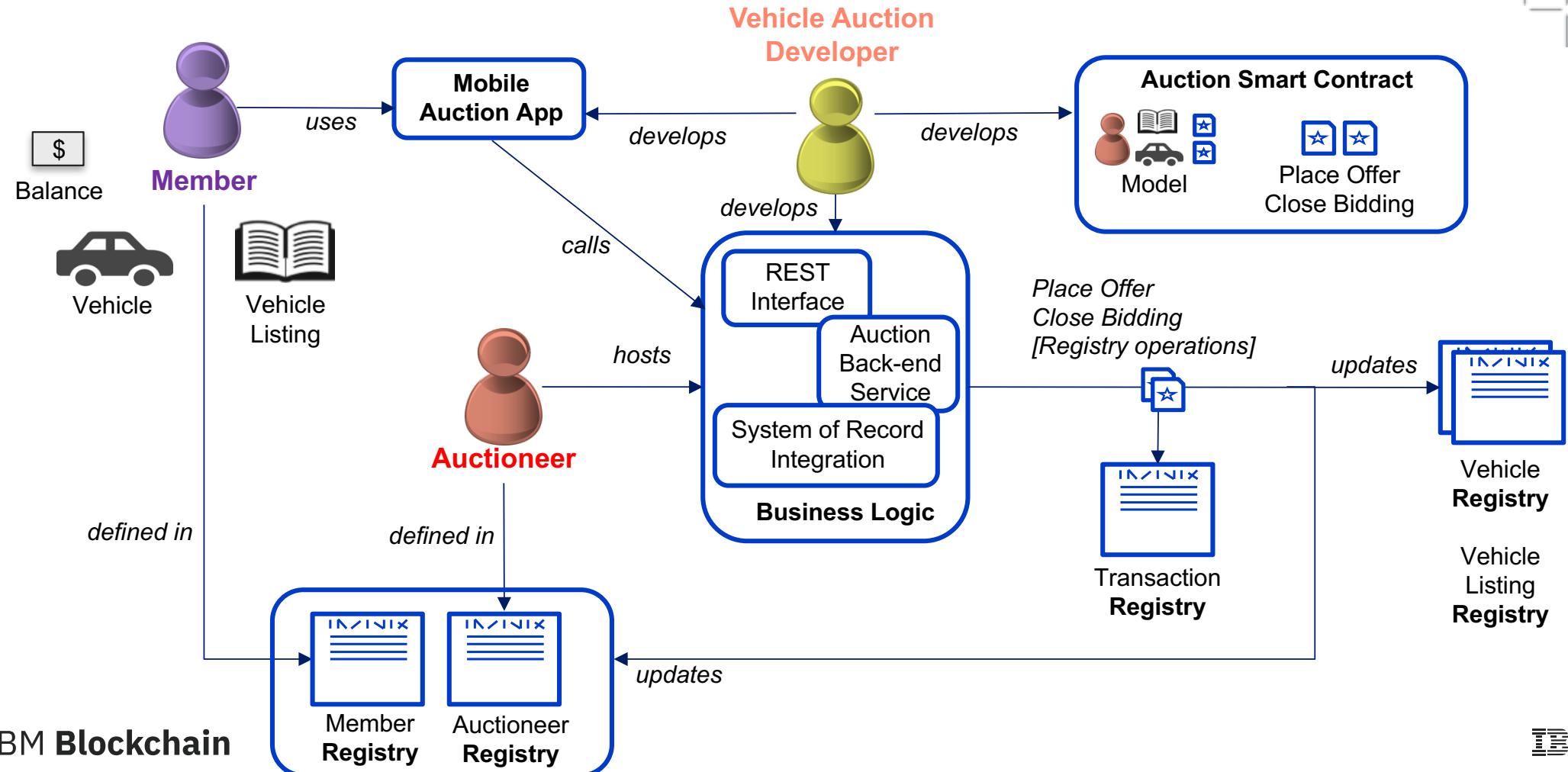
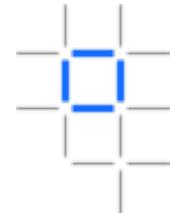
*Deploying to a blockchain
Interacting with systems of record*



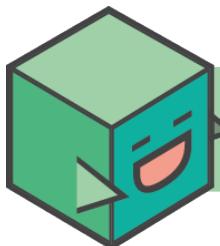
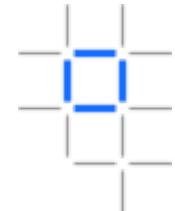
Key Concepts for the Business Service Provider



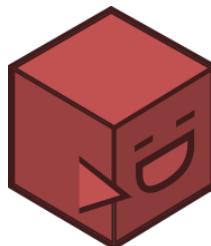
Key Concepts for a Vehicle Auction Developer



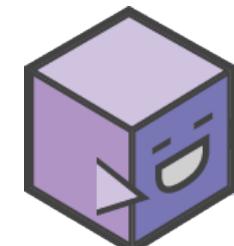
The Business Service Provider develops three components



Smart Contracts



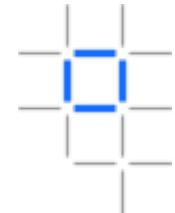
Business Logic



Presentation Logic

- Implements the logic deployed to the blockchain
 - **Models** describe assets, participants & transactions
 - **Transaction processors** provide the JavaScript implementation of transactions
 - **ACLs** define privacy rules
 - May also define events and registry queries
- **Services** that interact with the registries
 - Create, delete, update, query and invoke smart contracts
 - Implemented inside business applications, integration logic and REST services
 - Hosted by the Business Application Consumer
- Provides the **front-end** for the end-user
 - May be several of these applications
 - Interacts with business logic via standard interfaces (e.g. REST)
 - Composer can generate the REST interface from model and a sample application

Assets, Participants and Transactions

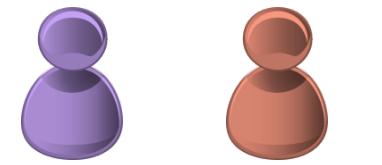


Vehicle



Vehicle Listing

```
asset Vehicle identified by vin {  
    o String vin  
    --> Member owner  
}  
  
asset VehicleListing identified by listingId {  
    o String listingId  
    o Double reservePrice  
    o String description  
    o ListingState state  
    o Offer[] offers optional  
    --> Vehicle vehicle  
}
```



Member Auctioneer

```
abstract participant User identified by email {  
    o String email  
    o String firstName  
    o String lastName  
}  
  
participant Member extends User {  
    o Double balance  
}  
  
participant Auctioneer extends User {  
}
```

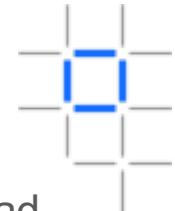


Place Offer
Close Bidding

```
transaction Offer {  
    o Double bidPrice  
    --> VehicleListing listing  
    --> Member member  
}  
  
transaction CloseBidding {  
    --> VehicleListing listing  
}
```

Transaction Processors

```
/**  
 * Close the bidding for a vehicle listing and choose the  
 * highest bid that is  
 * @param {org.acme.vehicle.auction.Offer} offer - the offer  
 * @transaction  
 */  
function closeBidding(  
    var listing = clos  
    if (listing.state  
        /**  
         * Make an Offer for a VehicleListing  
         * @param {org.acme.vehicle.auction.Offer} offer - the offer  
         * @transaction  
         */  
        function makeOffer(offer) {  
            var listing = offer.listing;  
            if (listing.state !== 'FOR_SALE') {  
                // Handle error  
            } else {  
                // Process bid  
            }  
        }  
    }  
}
```



Access Control

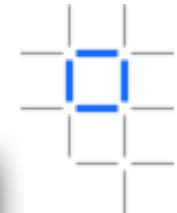
```
rule EverybodyCanReadEverything {
    description: "Allow all participants read access to all resources"
    participant: "org.acme.sample.SampleParticipant"
    operation: READ
    resource: "org.acme.sample.*"
    action: ALLOW
}
```

```
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
- 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

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

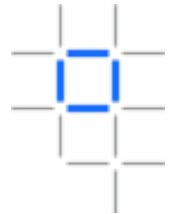
```
// Emit an event for the modified asset.  
var event = getFactory().newEvent('org.acme.sample', 'SampleEvent');  
event.asset = tx.asset;  
event.oldValue = oldValue;  
event.newValue = tx.newValue;  
emit(event);
```

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

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

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

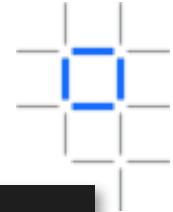
Smart Contract Development: Composer Playground



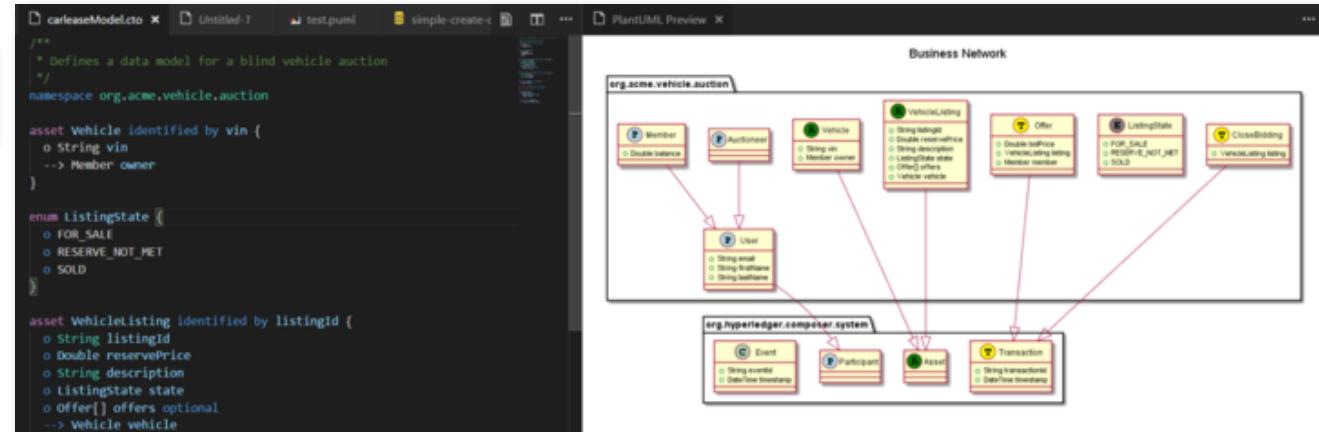
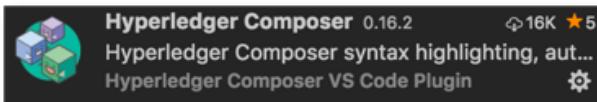
A screenshot of the Hyperledger Composer Playground web interface. The left sidebar shows files like 'About README.md', 'Model File models/sample.cto', 'Script File lib/sample.js', and 'Access Control permissions.acl'. The main area displays the 'Basic Sample Business Network' definition, which includes a 'Hello World' note, participant 'SampleParticipant', asset 'SampleAsset', transaction 'SampleTransaction', and event 'SampleEvent'. It also describes how SampleAssets are owned by a SampleParticipant and can be modified by submitting a SampleTransaction. A 'Test' tab is mentioned at the bottom.

- 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



The screenshot shows the Visual Studio Code interface with several tabs open:

- `carleaseModel.cto`: A Composer model file containing definitions for assets like `Vehicle` and `VehicleListing`, and an enum `ListingState`.
- `test.puml`: A PlantUML preview window showing a Business Network diagram for `org.acme.vehicle.auction`. It includes nodes for `Member`, `Auctioneer`, `Vehicle`, `User`, `Offer`, `ListingState`, and `VehicleListing`. Relationships show `Vehicle` being listed by `Auctioneer` and `Offer` being placed on `VehicleListing`.
- `simple-create-4`: Another tab showing some code.

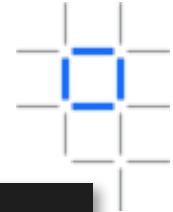
```
namespace org.acme.vehicle.lifecycle
```

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

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

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

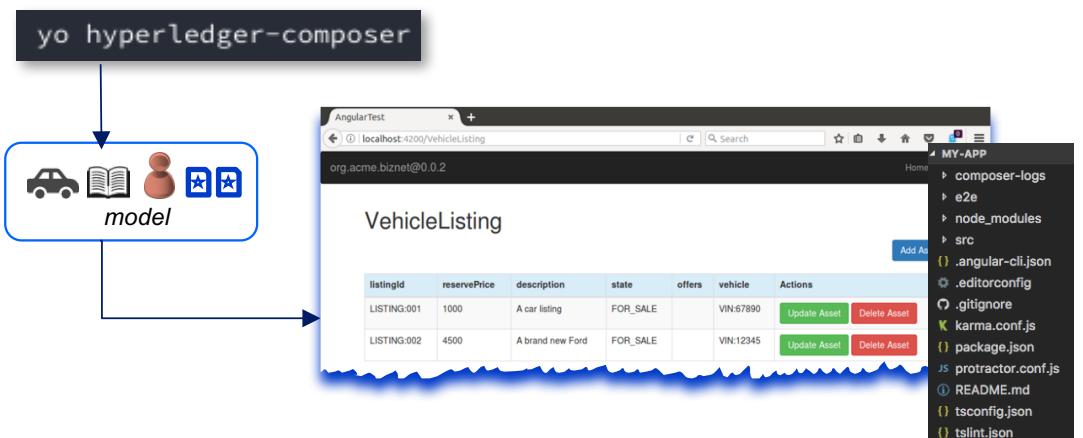
Creating the Business and End-User Applications



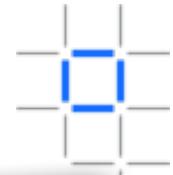
- JavaScript **business applications** require() the NPM “composer-client” module
 - This provides the API to access assets, participants and transactions
 - RESTful API (via Loopback) can also be generated... see later
- Command-line tool available to generate **end-user** command-line or Angular2 applications from model
 - Also helps with the generation of unit tests to help ensure quality code

```
const BusinessNetworkConnection = require('composer-client')
    .BusinessNetworkConnection;
this.bizNetworkConnection = new BusinessNetworkConnection();
this.titlesRegistry = this.bizNetworkConnection.getAssetRegistry
    ('net.biz.digitalPropertyNetwork.LandTitle')

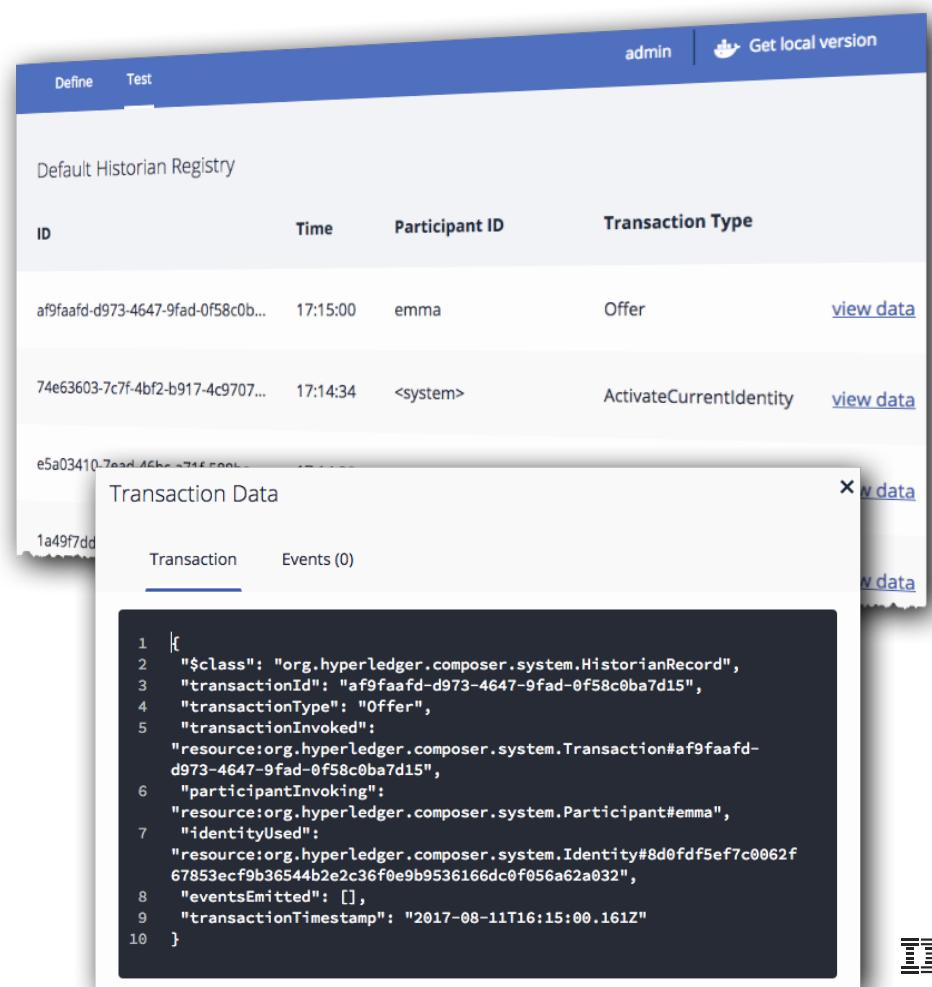
let landTitle1 = factory.newResource
    ('net.biz.digitalPropertyNetwork', 'LandTitle', 'LID:1148');
landTitle1.owner = ownerRelation;
landTitle1.information = 'A nice house in the country';
this.titlesRegistry.add(landTitle1);
```



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>



The screenshot shows the Hyperledger Composer Playground interface. At the top, there are tabs for 'Define' and 'Test', and a user 'admin'. A 'Get local version' button is also present. Below the tabs, the title 'Default Historian Registry' is displayed. A table lists three transactions:

ID	Time	Participant ID	Transaction Type	Action
af9faafed973-4647-9fad-0f58c0b...	17:15:00	emma	Offer	view data
74e63603-7c7f-4bf2-b917-4c9707...	17:14:34	<system>	ActivateCurrentIdentity	view data
e5a03410-7ead-46bc-a716-5901...	17:14:34	<system>	ActivateCurrentIdentity	view data

A modal window titled 'Transaction Data' is open for the first transaction (ID: af9faafed973-4647-9fad-0f58c0b...). It has tabs for 'Transaction' and 'Events (0)'. The 'Transaction' tab displays the following JSON code:

```
1  [
2    "$class": "org.hyperledger.composer.system.HistorianRecord",
3    "transactionId": "af9faafed973-4647-9fad-0f58c0b...d15",
4    "transactionType": "Offer",
5    "transactionInvoked": "resource:org.hyperledger.composer.system.Transaction#af9faafed973-4647-9fad-0f58c0b...d15",
6    "participantInvoking": "resource:org.hyperledger.composer.system.Participant#emma",
7    "identityUsed": "resource:org.hyperledger.composer.system.Identity#8d0fdf5ef7c0062f67853ecf9b36544b2e2c36f0e9b9536166dc0f056a62a032",
8    "eventsEmitted": [],
9    "transactionTimestamp": "2017-08-11T16:15:00.161Z"
10 }
```

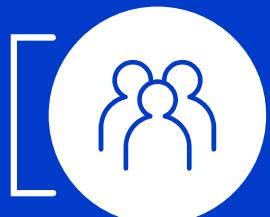


What is Hyperledger Composer?



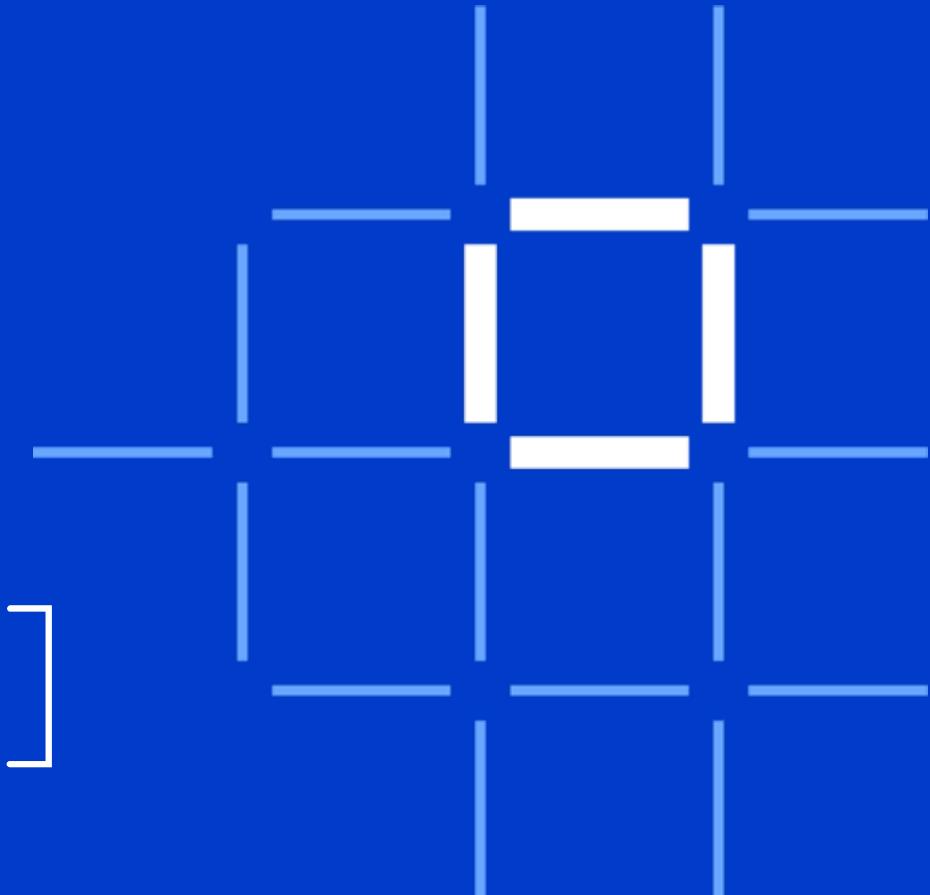
Application Development

*Writing the application
Modeling the business network*



Effective Administration

*Deploying to a blockchain
Interacting with systems of record*



There are Two User Roles with “Administration” Responsibility



- 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

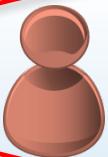
- Represents an organization on the business network



- 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

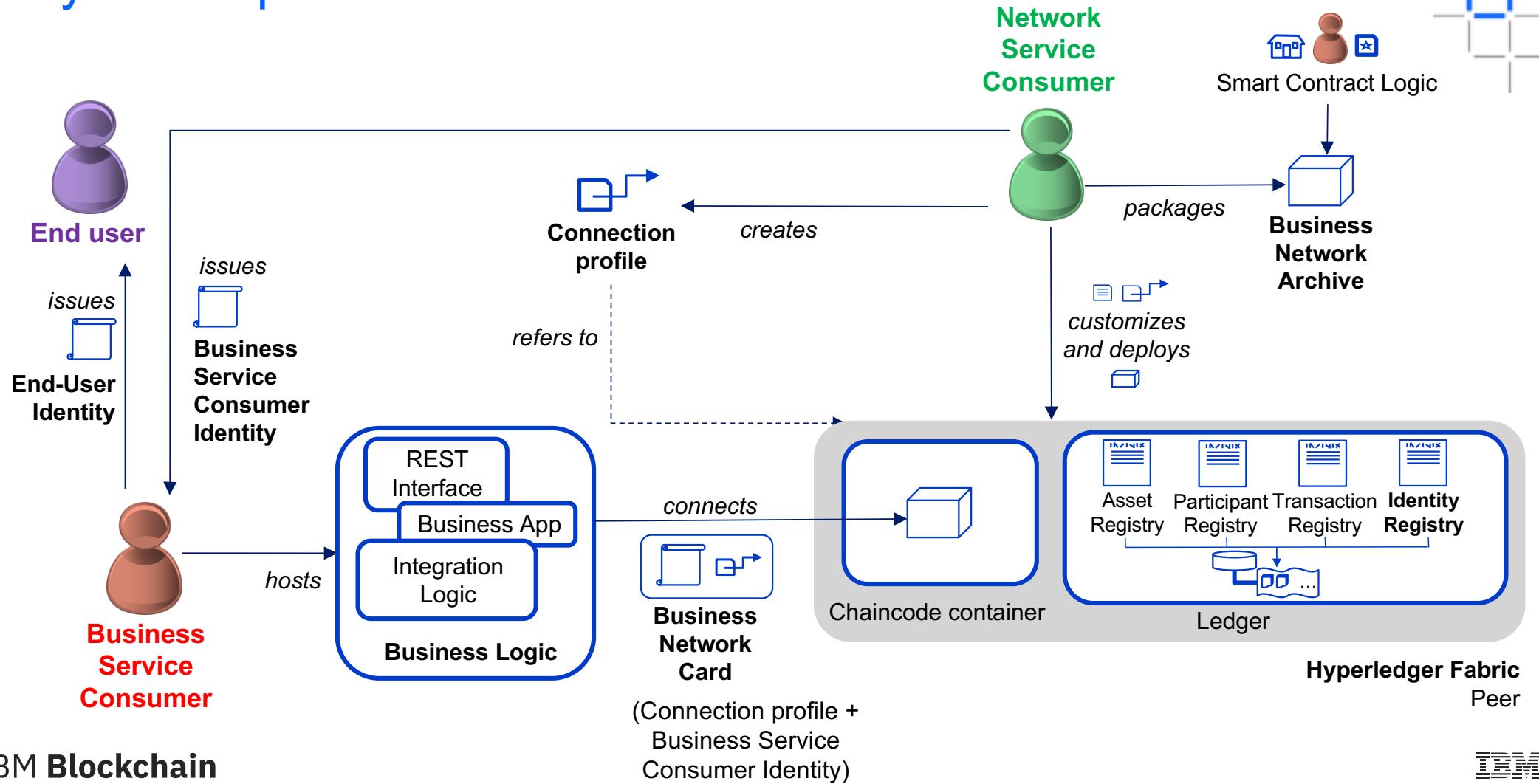
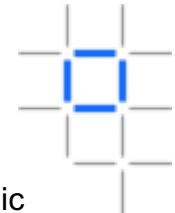


- End-user

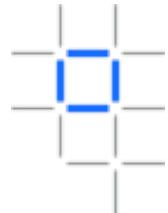
- Runs presentation logic e.g. on mobile device or dashboard

A single organization may play multiple roles!

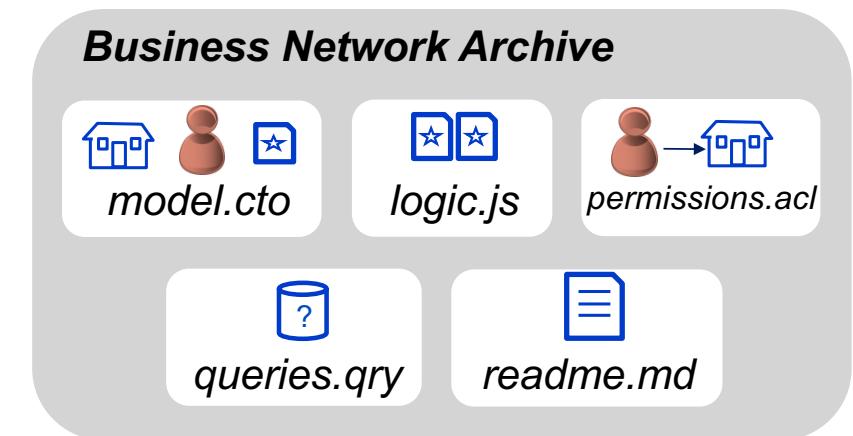
Key Concepts for Administrators



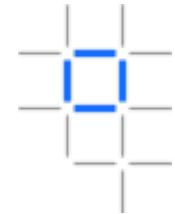
The Network Service Consumer packages resources into a BNA file



- Business Network Archive (.BNA) is a package of the resources used by Fabric:
 - Model files (.CTO)
 - Transaction processors (.JS)
 - Access Control Lists (.ACL)
 - Static queries (.QRY)
 - Documentation and versioning (.MD)
 - It does *not* contain the client application
- The BNA simplifies deployment of blockchain and promotion between environments
 - c.f. TAR, WAR, EAR, JAR, BAR...
- Create BNA files from Playground or command line
 - Build from filesystem or NPM module



```
composer archive create -archiveFile my.bna  
--sourceType module --sourceName myNetwork
```



Connection Profiles to Hyperledger Fabric

The screenshot shows the 'Basic Configuration' section of a connection profile named 'hifabric'. It includes fields for Orderer(s), Channel, MSP ID, Certificate Authority (CA), Peer(s), and Key Value Store Directory. A modal window displays the 'connection.json' file content:

```
1  {
2     "type": "hlfv1",
3     "orderers": [
4         { "url": "grpc://localhost:7050" }
5     ],
6     "ca": { "url": "http://localhost:7054",
7             "name": "ca.org1.example.com" },
8     "peers": [
9         {
10            "requestURL": "grpc://localhost:7051",
11            "eventURL": "grpc://localhost:7053"
12        }
13    ],
14    "keyValStore": "${HOME}/.composer-credentials",
15    "channel": "composerchannel",
16    "mspID": "Org1MSP",
17    "timeout": "300"
18 }
```

Buttons at the bottom include 'Use this profile' and 'Export Connection Profile'.

- 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

The image contains two screenshots of the IBM Blockchain Platform interface. The top screenshot shows the 'Issue New Identity' dialog box. It has fields for 'ID Name*' (set to 'emma_id'), 'Participant*' (set to 'resource:org.acme.vehicle.auction.Member#emma'), and a checkbox for 'Allow this ID to issue new IDs'. A note below says 'Issuing an identity generates a one-time secret. You can choose to send this to somebody or use it yourself when it has been issued.' The bottom screenshot shows the 'Identity Issued' card for the user 'emma_id'. It displays the connection profile 'hf1v1', user ID 'emma_id', and the business network 'carauction-network'. There are sections for 'Use it yourself' (with a 'Add to wallet' button) and 'Send it to someone else'. A note at the bottom says 'For security, new identities can only be enrolled once'.

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

Hyperledger Composer Playground

My Business Networks

Connection: hlfv1

Import Business Network Card Create Business Network Card

P PeerAdmin@hlfv1
USER ID: PeerAdmin
BUSINESS NETWORK: none
Connect now →

A myadmincard
USER ID: admin
BUSINESS NETWORK: carauction-network
Connect now →

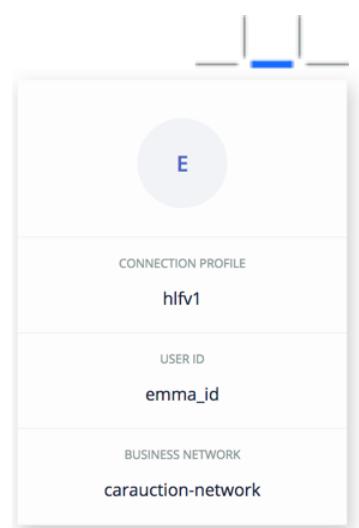
carauction-network
Deploy a new business network

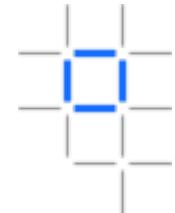
IBM Blockchain

- 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, command-line or from within your 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)

IBM Blockchain

angular-app

Auctioneer : A participant named Auctioneer	Show/Hide List Operations E
CloseBidding : A transaction named CloseBidding	Show/Hide List Operations E
Member : A participant named Member	Show/Hide List Operations E
Offer : A transaction named Offer	Show/Hide List Operations E
Vehicle : An asset named Vehicle	Show/Hide List Operations E
GET /Vehicle	Find all instances of the model matched by filter fro
POST /Vehicle	Create a new instance of the model and persist it in
GET /Vehicle/{id}	Find a model instance by {{id}} fro

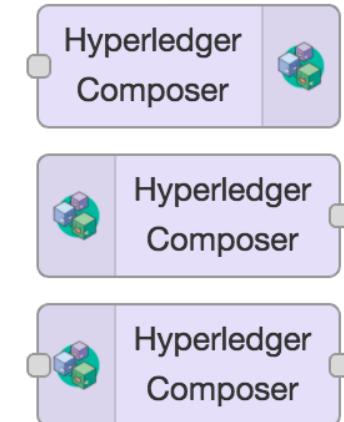
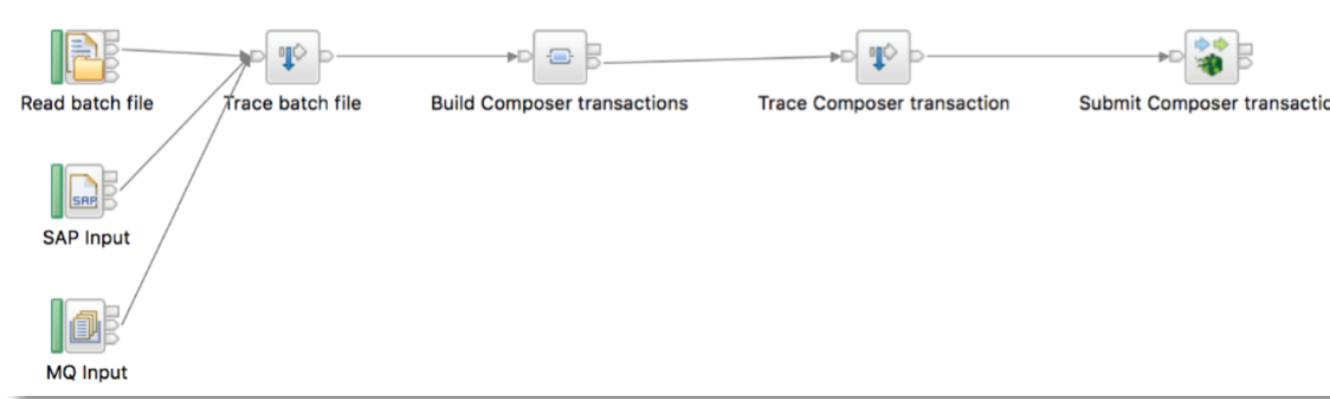
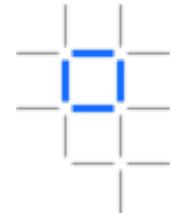
Request URL
`http://0.0.0.0:3000/api/Vehicle`

Response Body

```
[  
 {  
   "$class": "org.acme.vehicle.auction.Vehicle",  
   "vin": "VEH:1234",  
   "owner": "odowda@uk.ibm.com"  
 }  
]
```



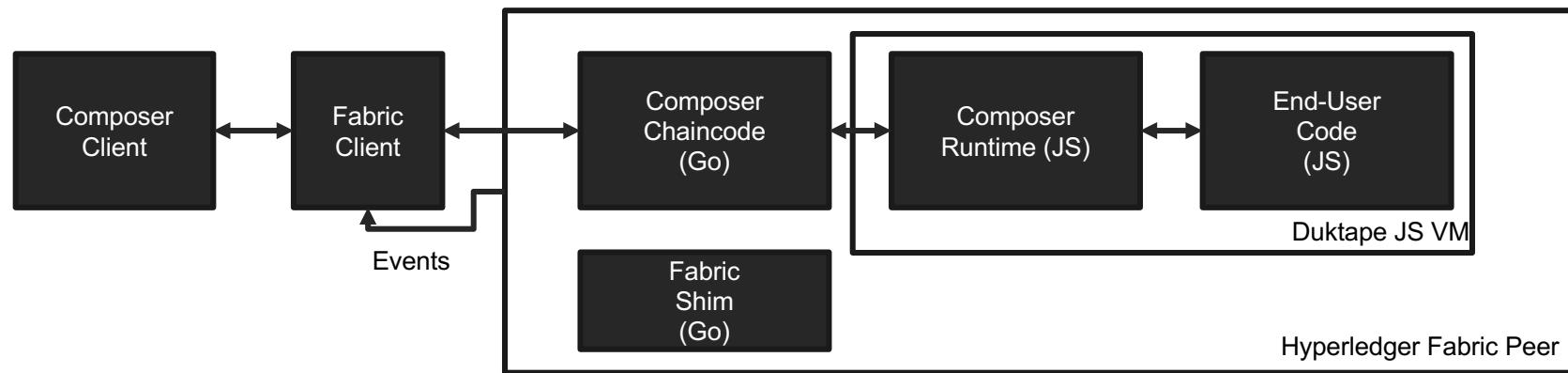
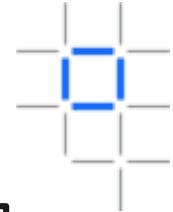
Exploiting Loopback: Examples



- **IBM Integration Bus**
 - IIB V10 contains Loopback connector
 - Example above takes input from file, SAP or MQ
 - Data mapping from CSV, BAPI/IDOC or binary form to JSON model definition

- **Node.RED**
 - Pre-built nodes available for Composer
 - Connect to hardware devices, APIs and online services
 - Install direct from Node.RED UI
 - Manage Palette -> Install -> node-red-contrib-composer

How Composer Maps to Fabric Chaincode



- Each Business Network is deployed to its own chaincode container
 - Container contains a static piece of Go chaincode that starts a Javascript virtual machine running transaction processors
- Browse these containers to view diagnostic information (docker logs)
- Embedded chaincode is not a Composer external interface

Hyperledger Composer Outlook

- Still early in product lifecycle
- Lots of improvements planned
 - See <https://github.com/hyperledger/composer/issues>
- An active development community
 - Open community calls every two weeks
 - Rocket Chat
 - Stack Overflow
- Get involved!

Hyperledger Rocket.Chat

You will need a [Linux Foundation ID](#), or alternatively you can log in with Facebook, GitHub, Google, or OpenStack.

[Let's chat](#)

Stack Overflow

Ask questions in Stack Overflow with the tag `#hyperledger-composer`.

[Ask now](#)

Contribute to the Project

GitHub

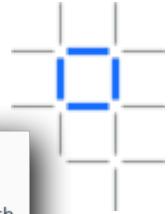
Check out the code, feel free to get involved.

[GitHub](#)

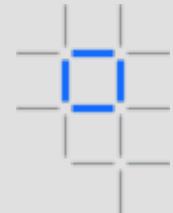
Community Call

Join our weekly open community calls.

[Learn how](#)



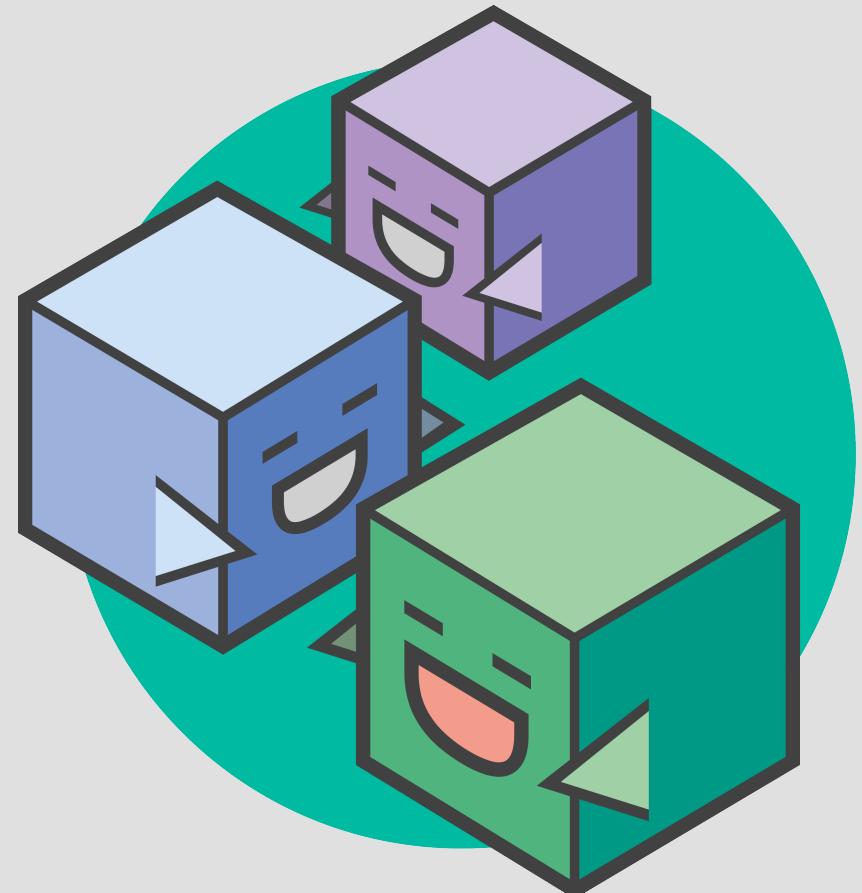
Get started with Hyperledger Composer



- Define, Test and Deploy Business Networks
- Create domain APIs and sample applications
- Integrate existing systems and data

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

<http://composer-playground.mybluemix.net/>



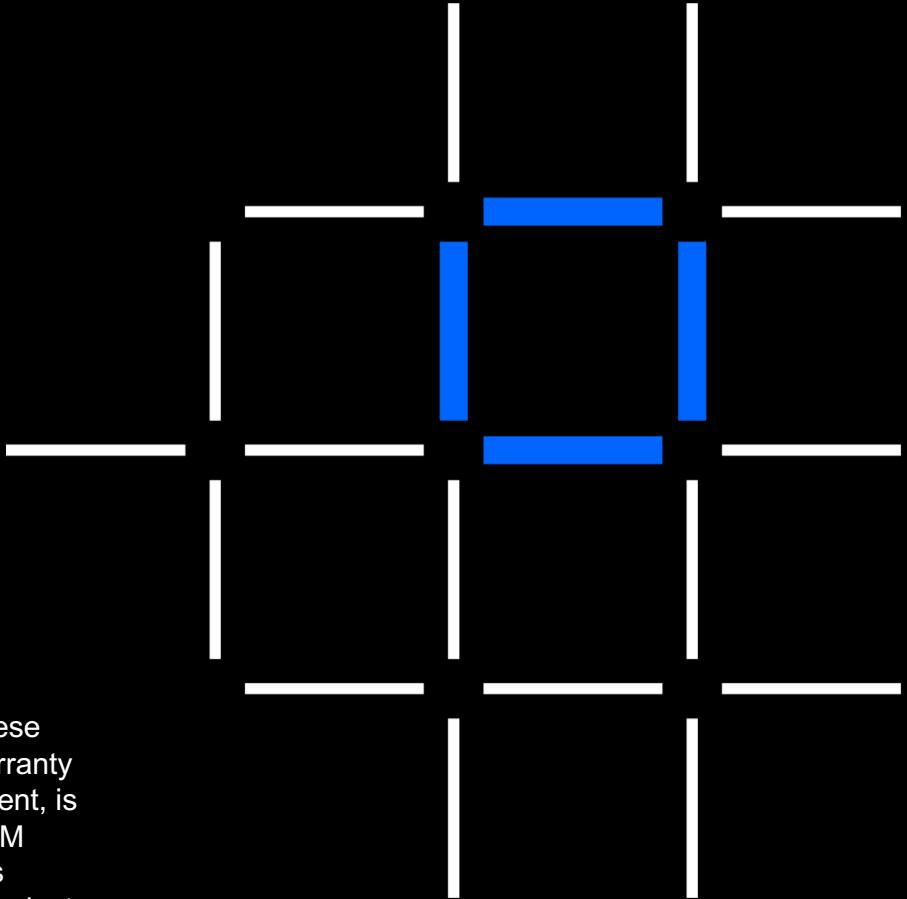
Thank you

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.

IBM

