

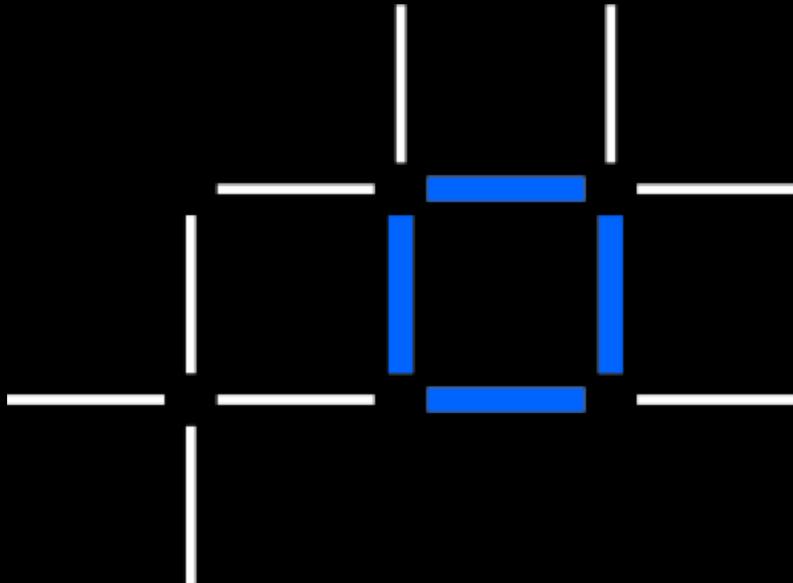
Blockchain Composed

A Technical Introduction to Hyperledger Composer

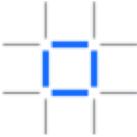
Austin Grice

austin.grice@ibm.com

Blockchain Leader



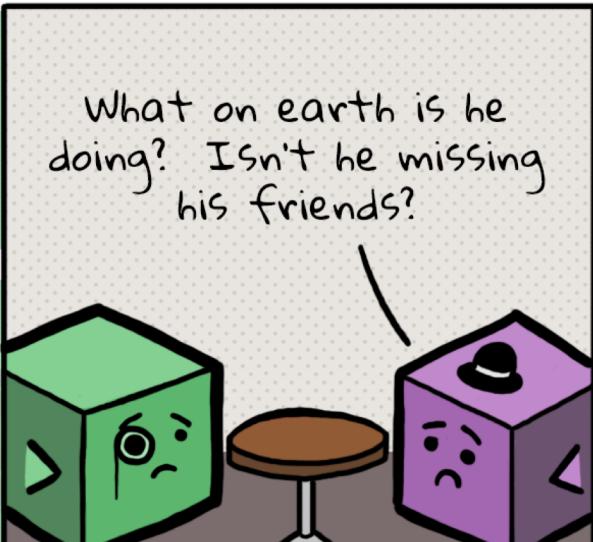
Blockchain Composed

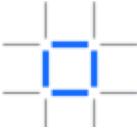


#M

CONGA COMICS

Block Height 20: "A cold one"

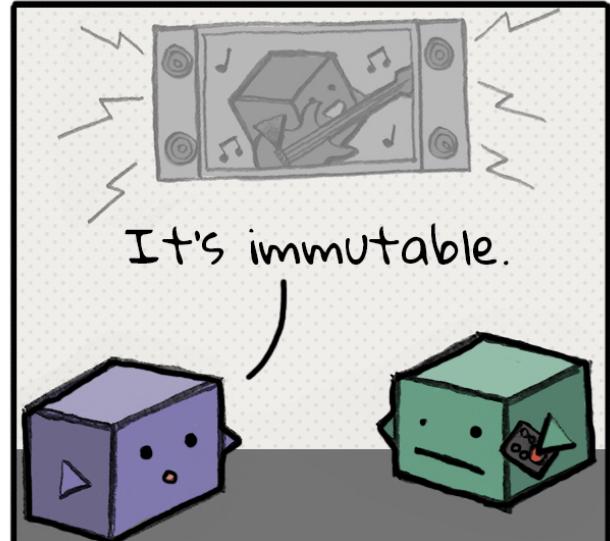
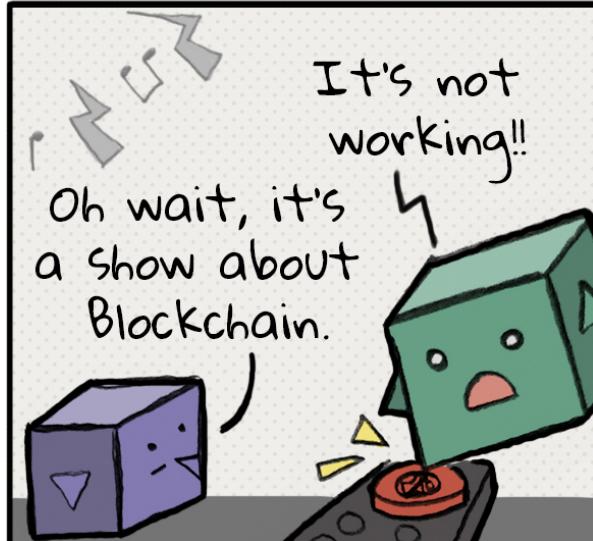
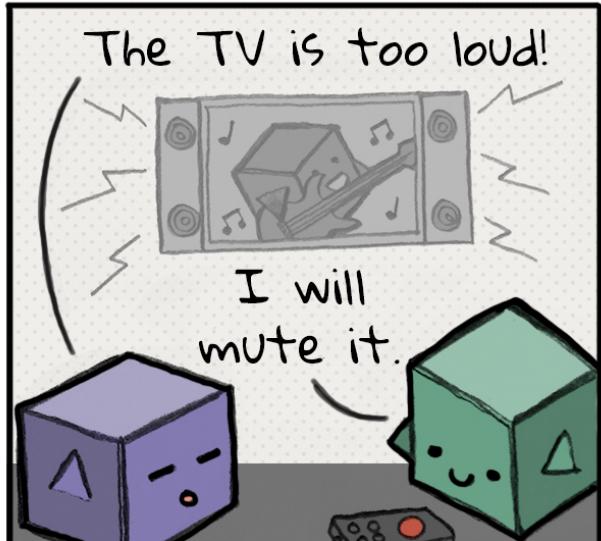


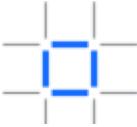


#M

CONGA COMICS

Block Height 1: "Loud Noises"

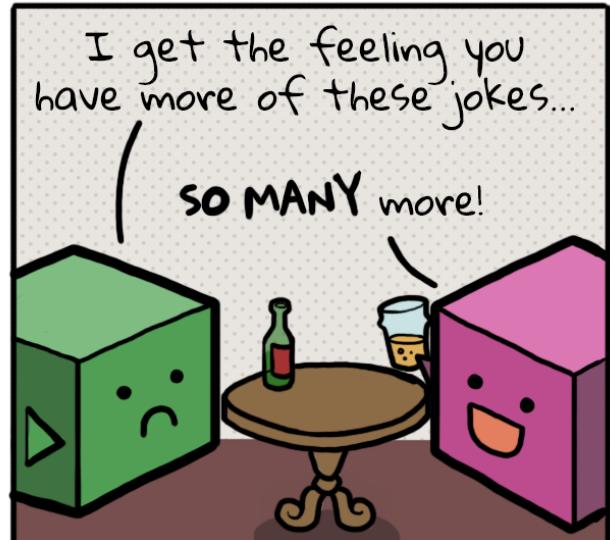
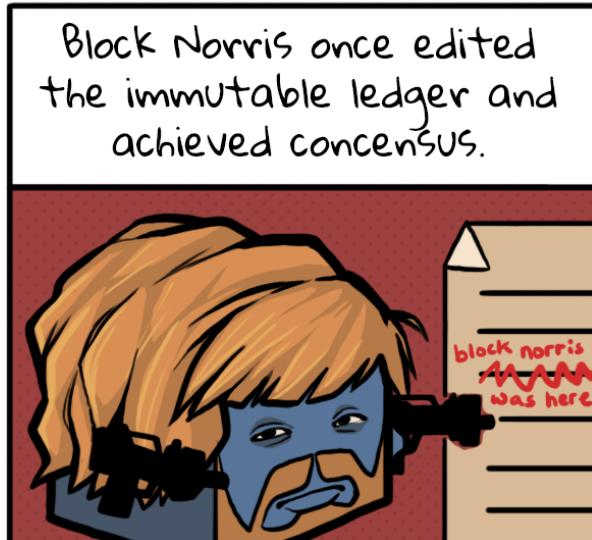
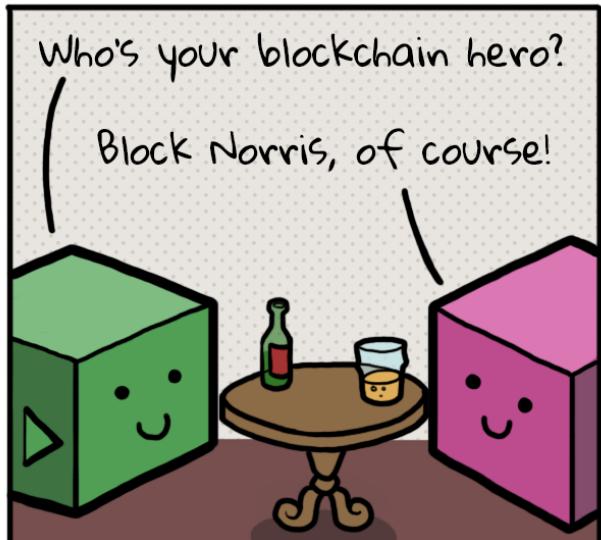


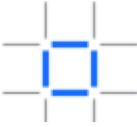


#M

CONGA COMICS

Block Height 15: "Enter Block Norris"

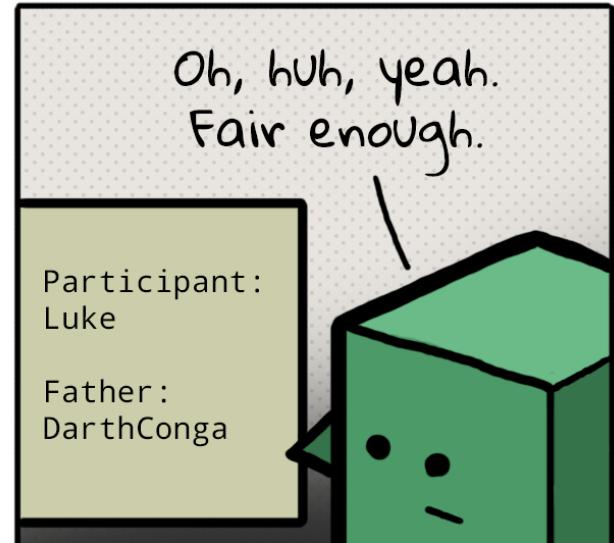




CONGA COMICS

Block Height 13: "Inheritance"

#M





What is Hyperledger Composer?



Application Development

Writing the application

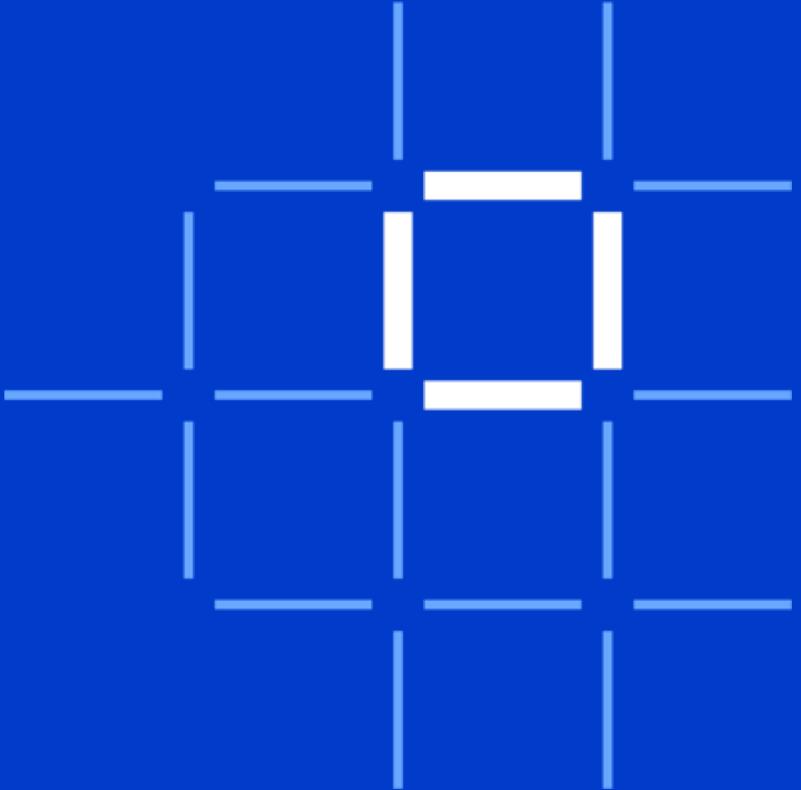
Modeling the business network



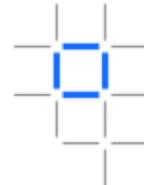
Effective Administration

Deploying to a blockchain

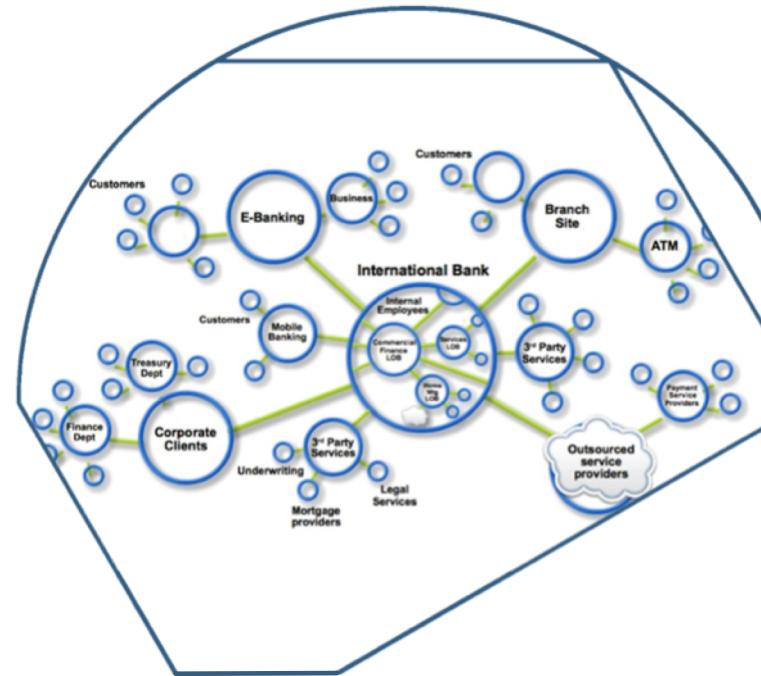
Interacting with systems of record



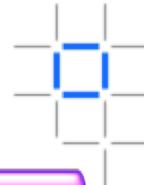
Blockchain Recap



- 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

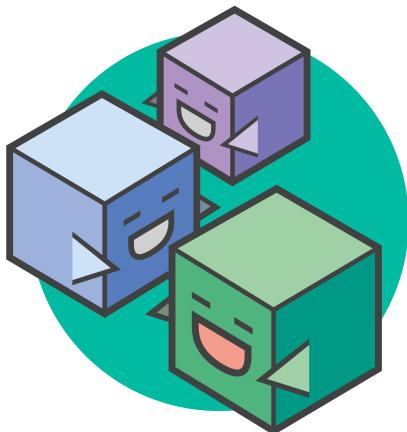


Hyperledger Composer: Accelerating time to value

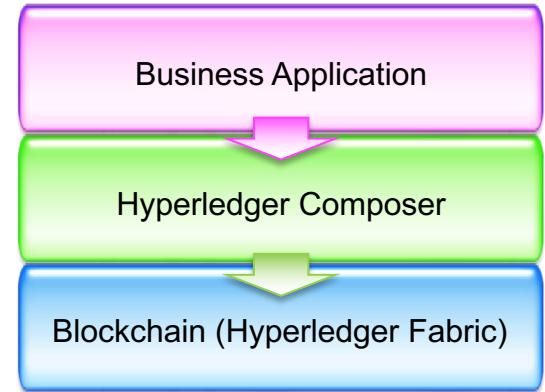


<https://hyperledger.github.io/composer/>

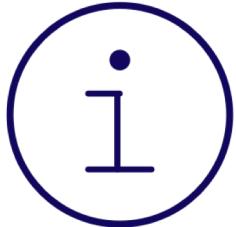
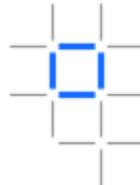
- A suite of high level application **abstractions** for business networks
- Emphasis on business-centric vocabulary for quick solution creation
- Reduce risk, and increase understanding and flexibility



- 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/>

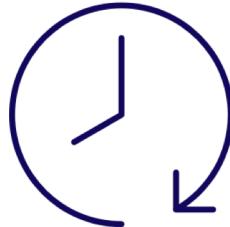


Benefits of Hyperledger Composer



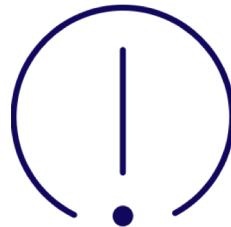
**Increases
understanding**

Bridges simply from
business concepts to
blockchain



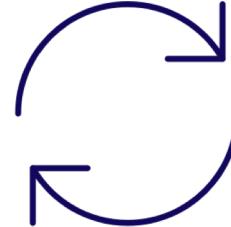
**Saves
time**

Develop blockchain
applications more
quickly and cheaply



**Reduces
risk**

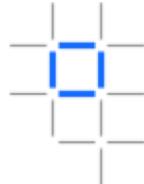
Well tested, efficient
design conforms to
best practice



**Increases
flexibility**

Higher level
abstraction makes it
easier to iterate

Extensive, Familiar, Open Development Toolset



```
asset Animal identi  
  o String animal]  
  o AnimalType sp  
  o MovementStatus  
  o ProductionTyp
```

Data modelling

A yellow square containing the letters "JS" in black.

JavaScript
business logic



Web playground

```
composer-client  
composer-admin
```

The npm logo, which consists of the lowercase letters "npm" in a red sans-serif font.

Client libraries



Editor support

```
$ composer
```

CLI utilities



Code generation

Powered by
 LoopBack
Node.js Framework

A green circle containing three white curly braces "…".
Swagger

Existing systems and
data

User Roles in a Blockchain Project



- Network Provider

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



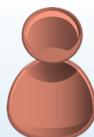
- Network Consumer

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



- Developer

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



- Business 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?



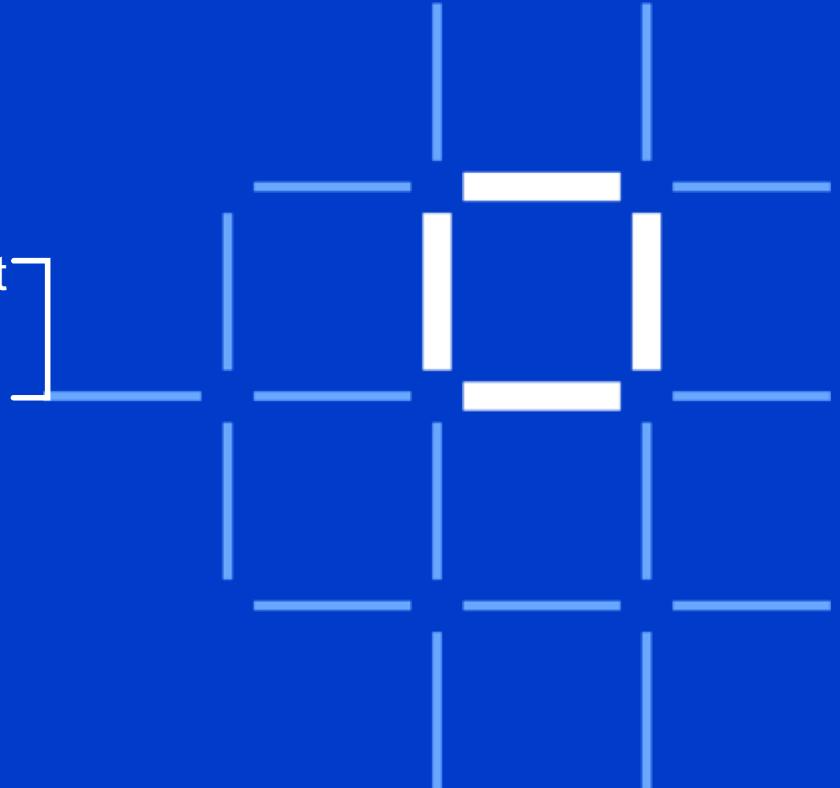
Application Development

*Writing the application
Modeling the business network*

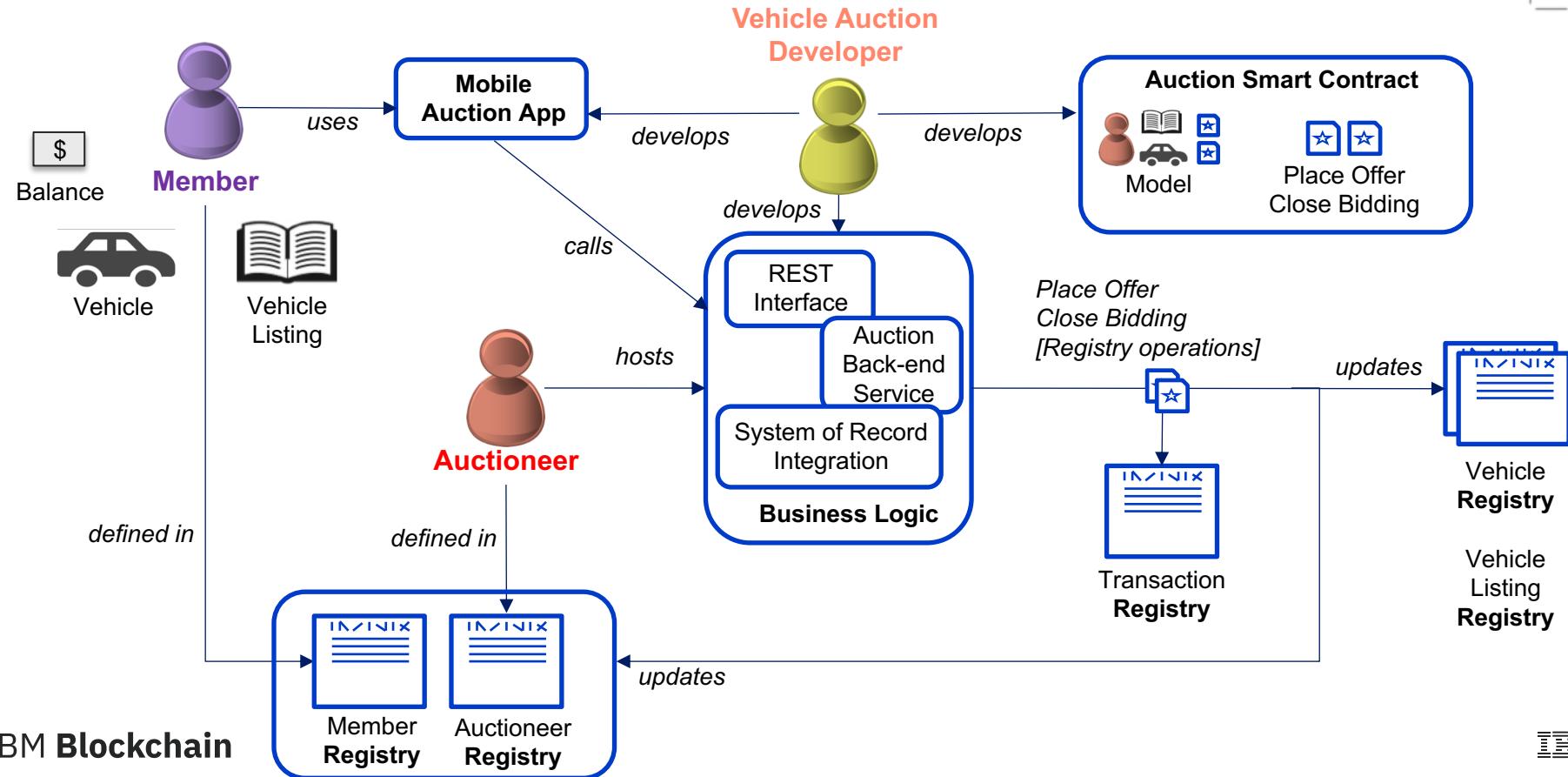
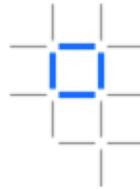


Effective Administration

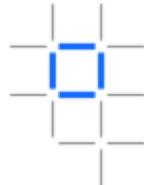
*Deploying to a blockchain
Interacting with systems of record*



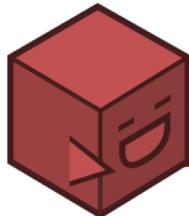
Key Concepts for a Vehicle Auction Developer



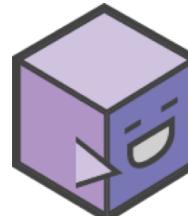
The Developer's three components



Smart Contracts



Business Logic



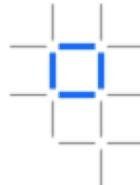
Presentation Logic

- 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 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

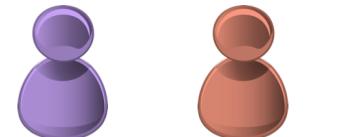


Vehicle



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 {  
}
```



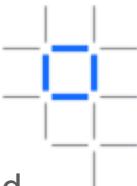
Place Offer
Close Bidding

```
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.listing} listing - the listing  
 * @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') {
```



Access Control

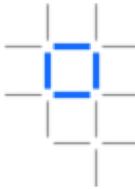
```
rule EverybodyCanReadEverything {
    description: "Allow all participants read access to all resources"
    participant: "org.acme.sample.SampleParticipant"
    operation: READ
    resource: "org.acme.sample.*"
    action: ALLOW
}
```

```
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 anything 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

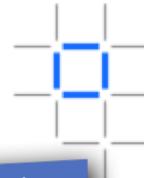
Smart Contract Development: Composer Playground



A screenshot of the Hyperledger Composer Playground web interface. The left sidebar shows files: README.md, models/sample.cto, lib/sample.js, and permissions.acl. A central panel displays the "Basic Sample Business Network". It includes a description of the "Hello World" sample, listing participants, assets, transactions, and events. It also describes how SampleAssets are owned by a SampleParticipant and can be modified. Below this, instructions for testing the network are provided, along with a note about creating a SampleParticipant participant. The bottom navigation bar includes links for Legal, GitHub, Tutorial, Docs, Community, and playground v0.16.3.

- 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 v1.1, or simulate completely within browser
- Install on your machine or run online at <http://composer-playground.mybluemix.net>

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
- 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' with a 'Get local version' button. Below this is a table titled 'Default Historian Registry' with columns: ID, Time, Participant ID, and Transaction Type. Two rows are listed:

ID	Time	Participant ID	Transaction Type
af9faaf9-d973-4647-9fad-0f58c0ba7d15	17:15:00	emma	Offer
74e63603-7cf-4bf2-b917-4c9707...	17:14:34	<system>	ActivateCurrentIdentity

A modal window titled 'Transaction Data' is open for the first row. It has tabs for 'Transaction' and 'Events (0)'. The 'Transaction' tab displays a JSON object:

```
1  {
2    "$class": "org.hyperledger.composer.system.HistorianRecord",
3    "transactionId": "af9faaf9-d973-4647-9fad-0f58c0ba7d15",
4    "transactionType": "Offer",
5    "transactionInvoked",
6    "resource:org.hyperledger.composer.system.Transaction#af9faaf9-d973-4647-9fad-0f58c0ba7d15",
7    "participantInvoking",
8    "resource:org.hyperledger.composer.system.Participant#emma",
9    "identityUsed",
10   "resource:org.hyperledger.composer.system.Identity#8d0fdf5ef7c0062f67853ecf9b36544b2e2c36f0e9b9536166dc0f056a62a032",
11   "eventsEmitted": [],
12   "transactionTimestamp": "2017-08-11T16:15:00.161Z"
13 }
```



What is Hyperledger Composer?



Application Development

Writing the application

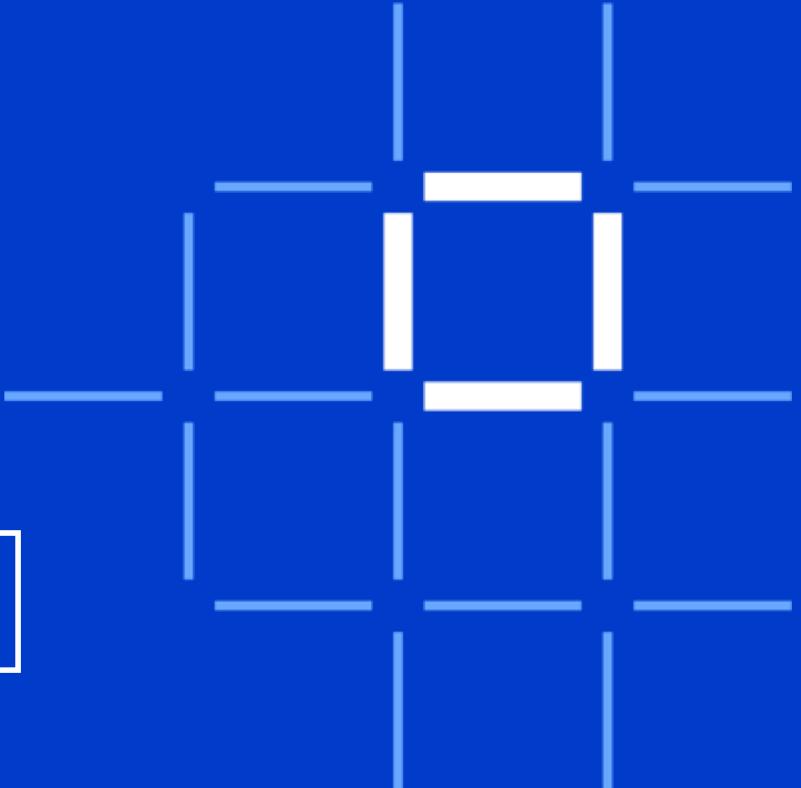
Modeling the business network



Effective Administration

Deploying to a blockchain

Interacting with systems of record



There are Two User Roles with “Administration” Responsibility



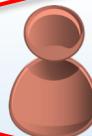
- Network Provider
 - **Governs** the network: channels, membership etc.
 - A consortium of network members or designated authority



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



- Developer
 - **Develops** blockchain business applications
 - Includes transaction, app server, integration and presentation logic



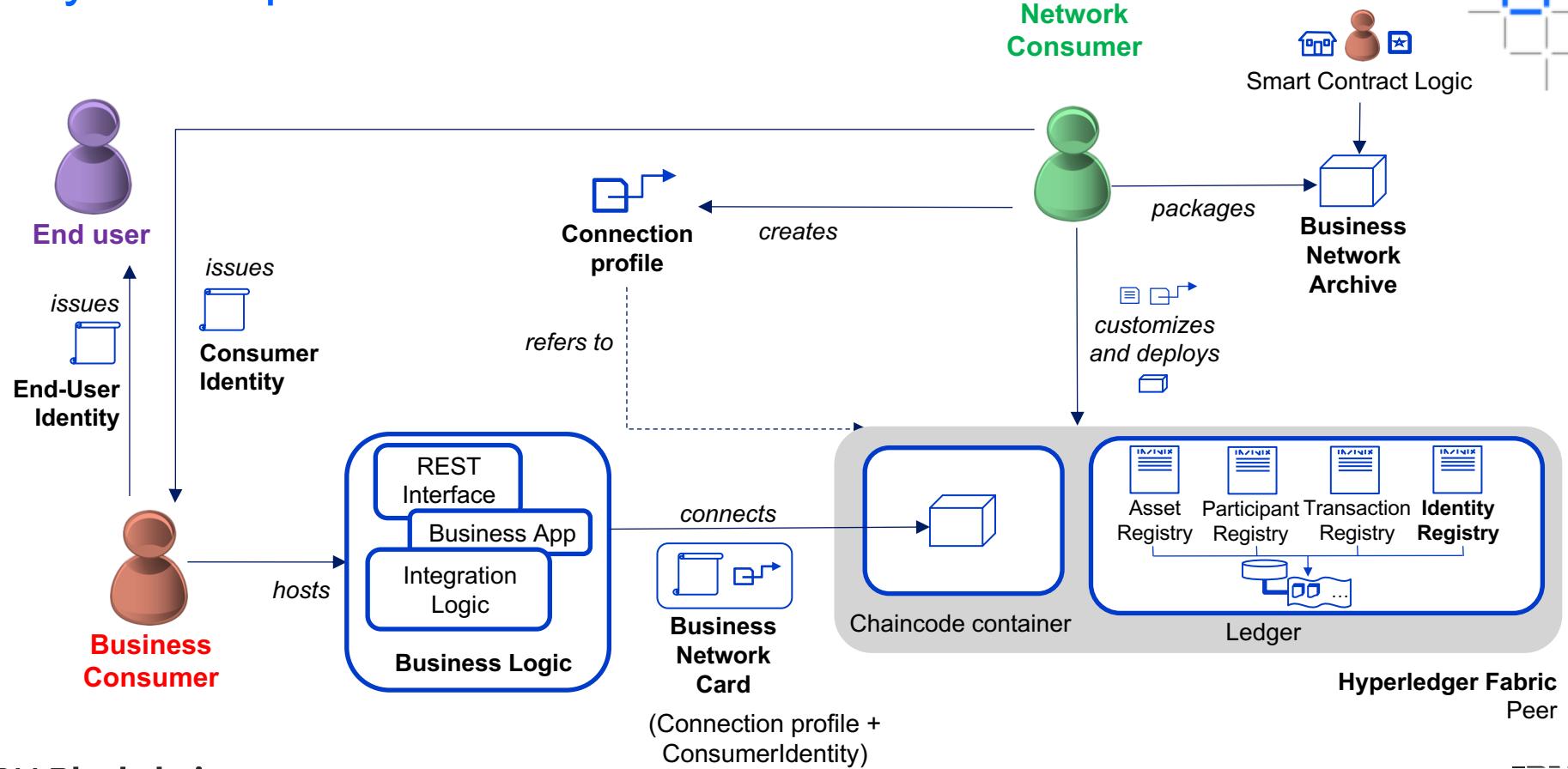
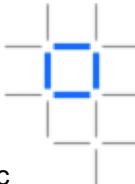
- Business Consumer
 - Hosts application and integration logic which invokes blockchain transactions

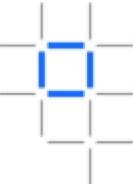


- End-user
 - Runs presentation logic e.g. on mobile device or dashboard

A single organization may play multiple roles!

Key Concepts for Administrators

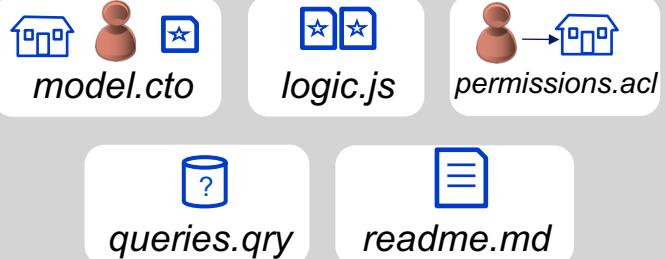




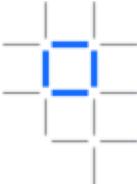
The Network Consumer packages resources into a BNA file

- 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

Business Network Archive



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



Connection Profiles to Hyperledger Fabric

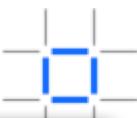
Basic Configuration

Connection Profile Name	hifabric
Orderer(s)	Orderer URL grpc://orderer.example.com:7050
Channel	composerchannel
MSP ID	Org1MSP
Certificate Authority (CA)	URL http://ca.org1.example.com:7054 Name ca.org1.example.com
Peer(s)	Peer Request URL grpc://peer0.org1.example.com:7051 Peer Event URL grpc://peer0.org1.example.com:7053
Key Value Store Directory	/home/composer/.composer-credentials

Advanced

Use this profile 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
 - Issue Fabric identity from Composer participants
- Connection profiles currently used by Composer and Fabric at v1.1



Participant Identity

- The **Network 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 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

Issue New Identity

Issue a new ID to a participant in your business network

ID Name* emma_id

Participant* resource:org.acme.vehicle.auction.Member#emma

Allow this ID to issue new IDs ()

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.

Identity Issued

E

CONNECTION PROFILE hlfv1

USER ID emma_id

BUSINESS NETWORK carauction-network

▼ Use it yourself
Just add the business network card to your wallet to start using the new identity yourself
Give the business network card a name
e.g. emma_id@carauction-network

➤ Send it to someone else

⚠ 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
 - Similar to an ATM card

Hyperledger Composer Playground

My Business Networks

Connection: hlfv1

Import Business Network Card Create Business Network Card

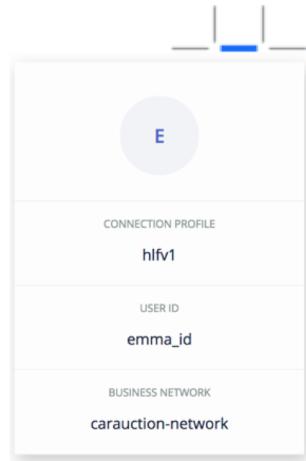
User ID	Business Network
PeerAdmin	none
admin	carauction-network

PeerAdmin@hlfv1 myadmincard

USER ID BUSINESS NETWORK

Connect now → Connect now →

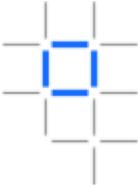
Deploy a new business network



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

```
composer network deploy -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)

angular-app

Auctioneer : A participant named Auctioneer

Show/Hide | List Operations |

CloseBidding : A transaction named CloseBidding

Show/Hide | List Operations |

Member : A participant named Member

Show/Hide | List Operations |

Offer : A transaction named Offer

Show/Hide | List Operations |

Vehicle : An asset named Vehicle

Show/Hide | List Operations |

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"  
  }  
]
```

IBM

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

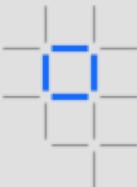
Check out the code, feel free to get involved.

[GitHub](#)

Community Call

Join our weekly open community calls.

[Learn how](#)

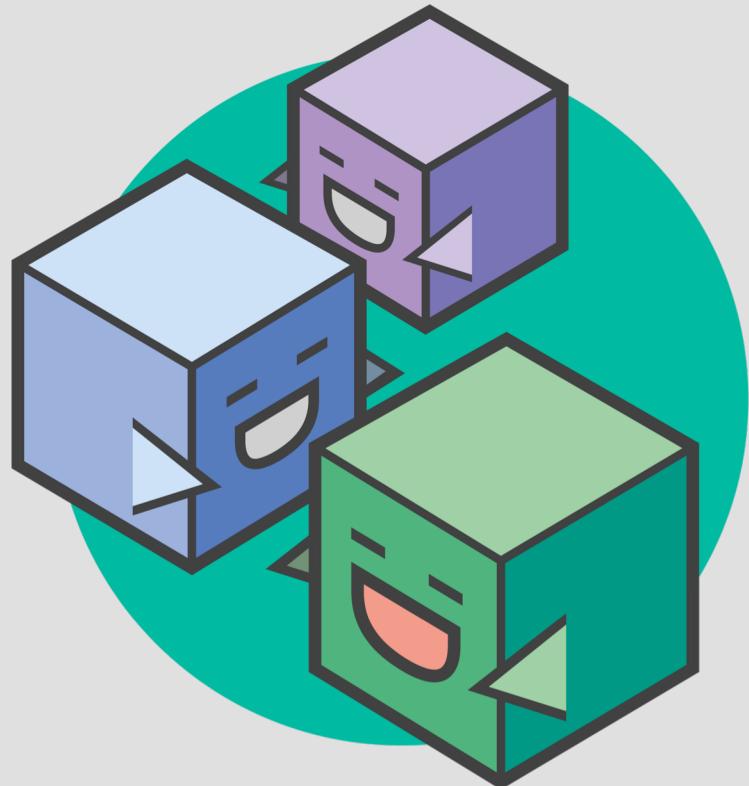


Get started with Hyperledger Composer

- 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

Austin Grice

austin.grice@ibm.com

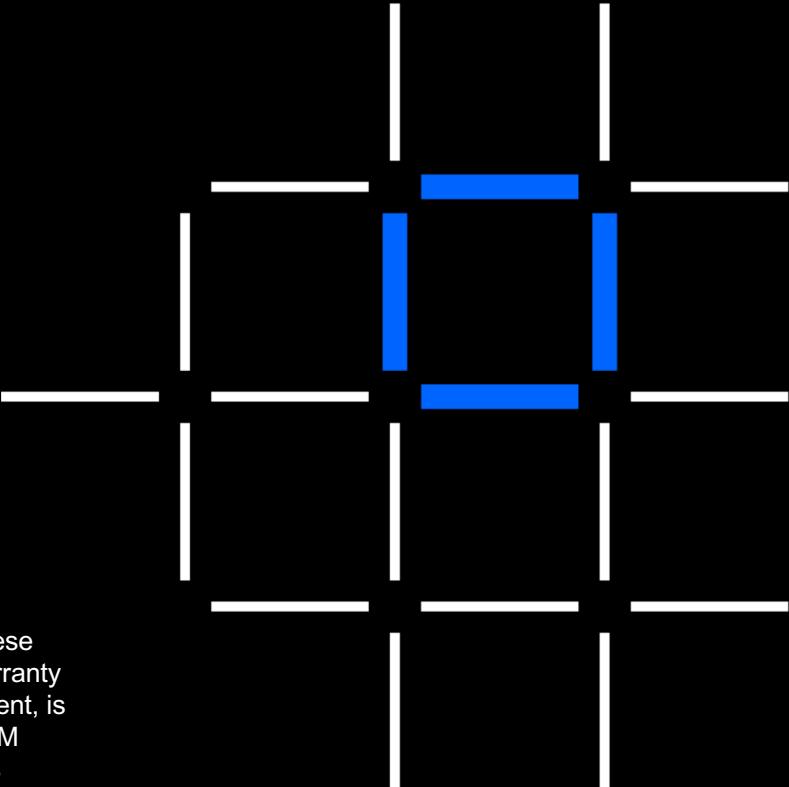
Blockchain Leader

IBM Blockchain

www.ibm.com/blockchain

developer.ibm.com/blockchain

www.hyperledger.org



© Copyright IBM Corporation 2018. 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.



