HYPERLEDGER

BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

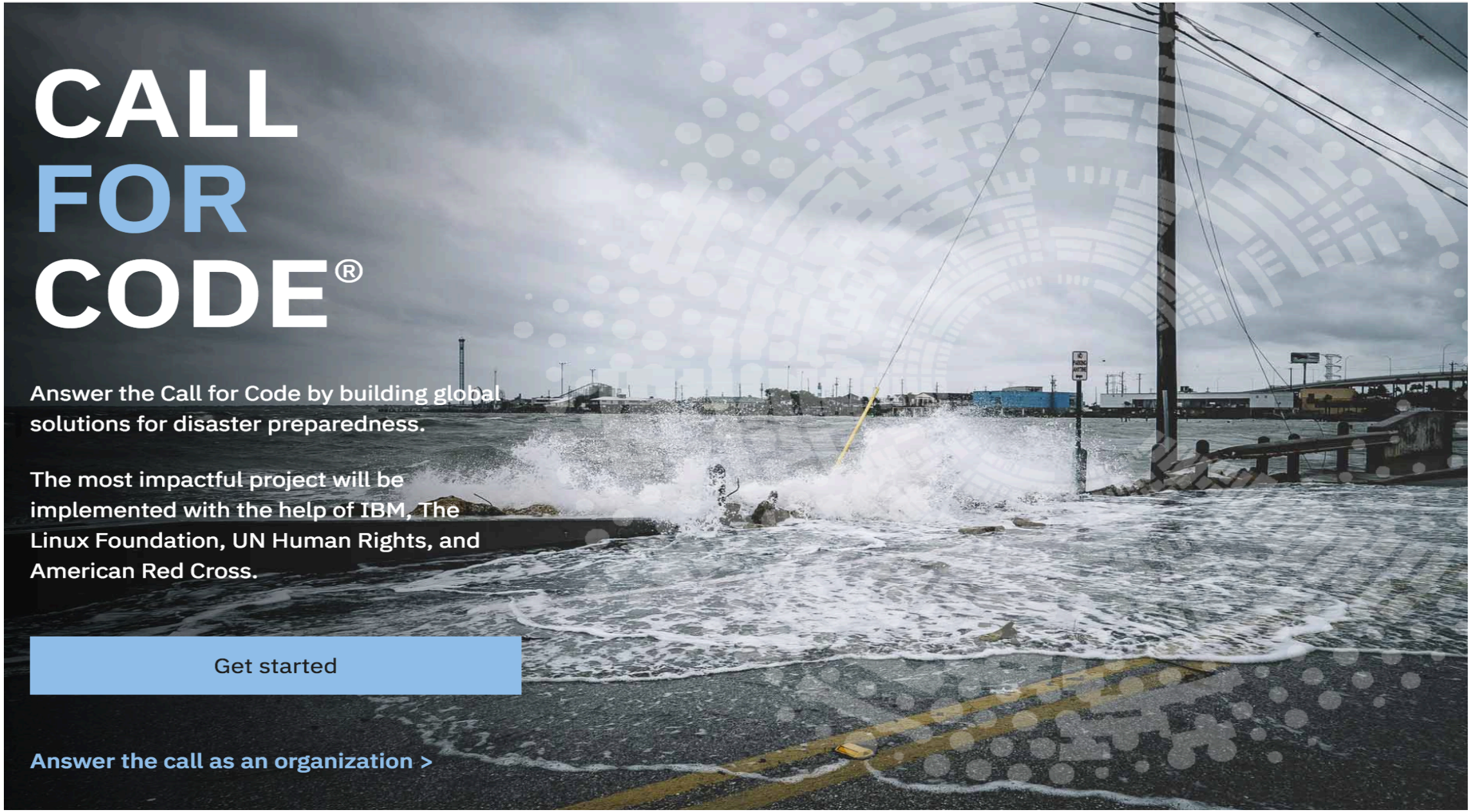
Creating Blockchain applications on Hyperledger Fabric and the IBM Cloud

Lennart Frantzell, IBM Developer Advocate

alf@us.ibm.com

August 2018

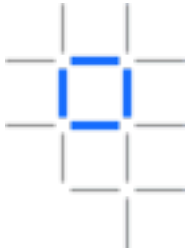
http://ibm.biz/portblock



IBM **Blockchain**

https://callforcode.org/

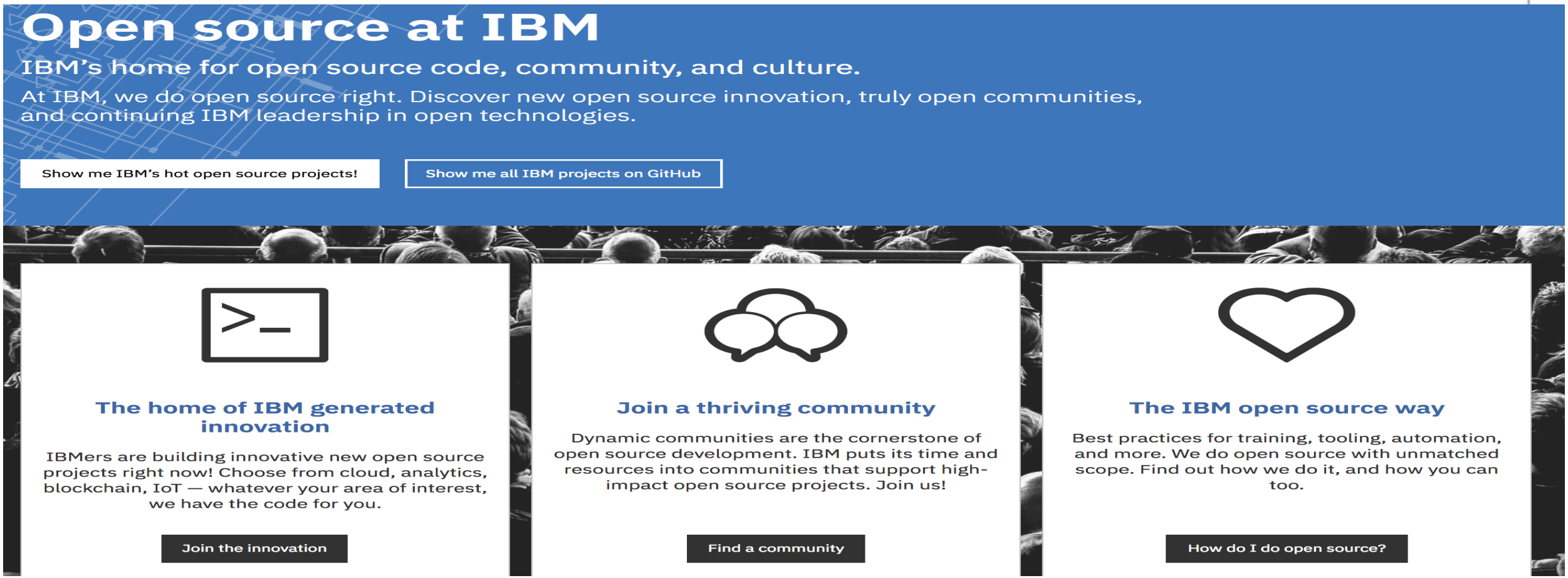
Blockchain and IBM



IBM's three most important technologies:

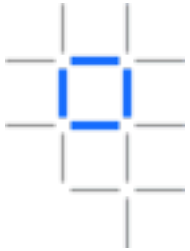
- Watson and AI
- Blockchain with a team of 1,500 people
- Internet of Things

IBM and the Blockchain Story



https://developer.ibm.com/code/open/

IBM and Blockchain, the story



1. **Our blockchain team of 1,500 has experience with more than 400 engagements around the globe to develop blockchain applications**, activate blockchain networks, and get the most value from this disruptive technology. For instance, we are working with:
2. **The U.S. Food and Drug Administration to study the use of blockchain technology for secure exchange of healthcare data**.
3. Sony and Sony Global Education to develop a new system, built on the IBM Blockchain platform, **to manage students' learning data**.
4. **we.Trade, a group of nine of Europe's largest banks, provides a new trade finance platform** designed to simplify domestic and cross-border trade for small and medium enterprises in Europe, while helping to increase overall trade transaction transparency.
5. Also, we recently announced that we are creating **a joint venture with shipping giant Maersk** that is meant to transform the global shipping industry.

IBM **Blockchain** http://newsroom.ibm.com/IBM-blockchain?item=30363

IBM and Maersk: Tradelens.com

https://tradelens.com/

everledger

Legitimize Diamonds and Reduce Fraud

What? Provenance.

- Track diamonds across supply chain from mine to retail

How?

- **Shared ledger for storing digital certification with supporting material**

Benefits

1. Protect against the occurrence of fraud, theft, trafficking and black markets
2. Assist in the identification and reduction of synthetic stones being labelled as authentic
3. Increase speed of transparency for cross border transactions for insurance companies, banks and claimants

Food Safety

What? Provenance

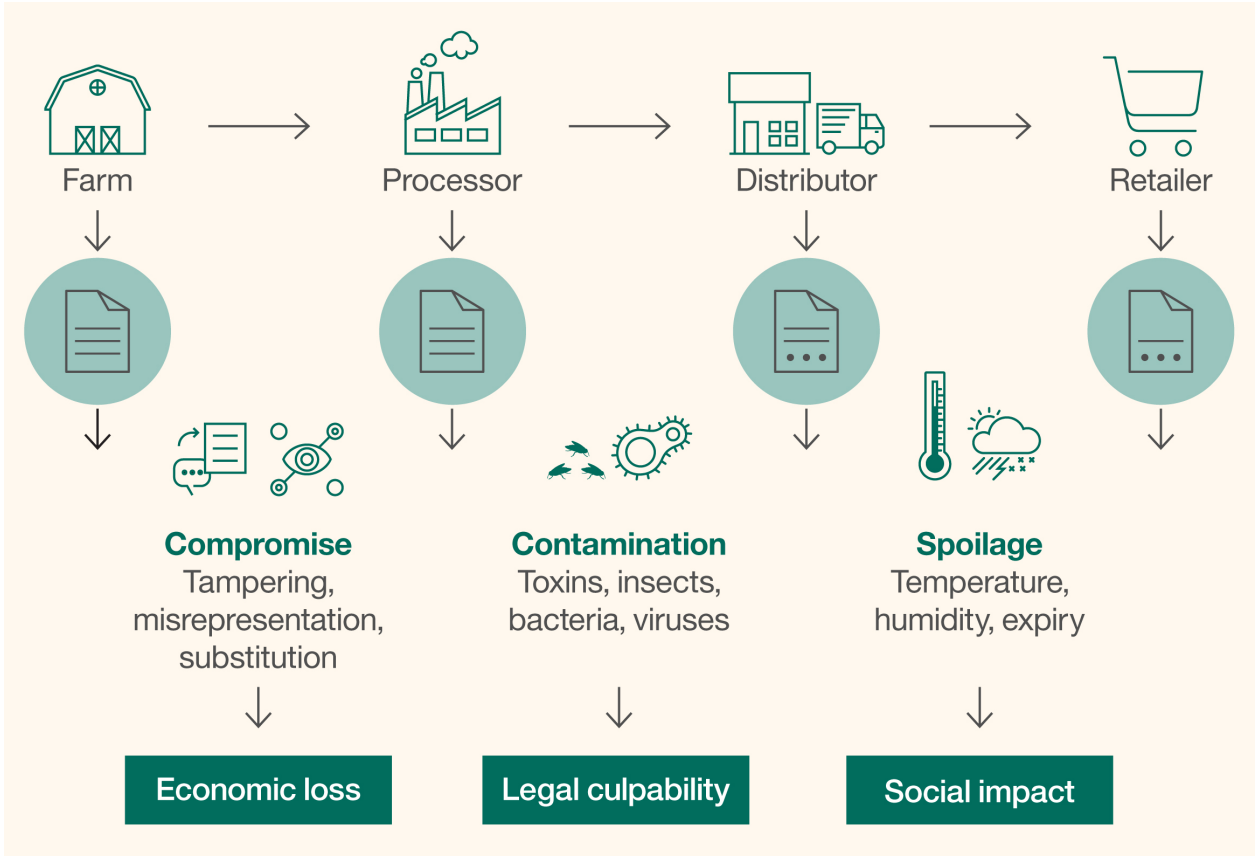
- Provide a trusted source of information and traceability to improve transparency and efficiency across the food network.

How?

- Shared ledger for storing digital compliance documentation, test results and audit certificates

Benefits

- Reduce impact of food recalls through instant access to end-to-end traceability data to verify history in the food network and supply chain.
- Help to address the 1 in 10 people sickened and 400,000 fatalities WW which occur every year from food-born illnesses.



Global Financing: Dispute Resolution

What?

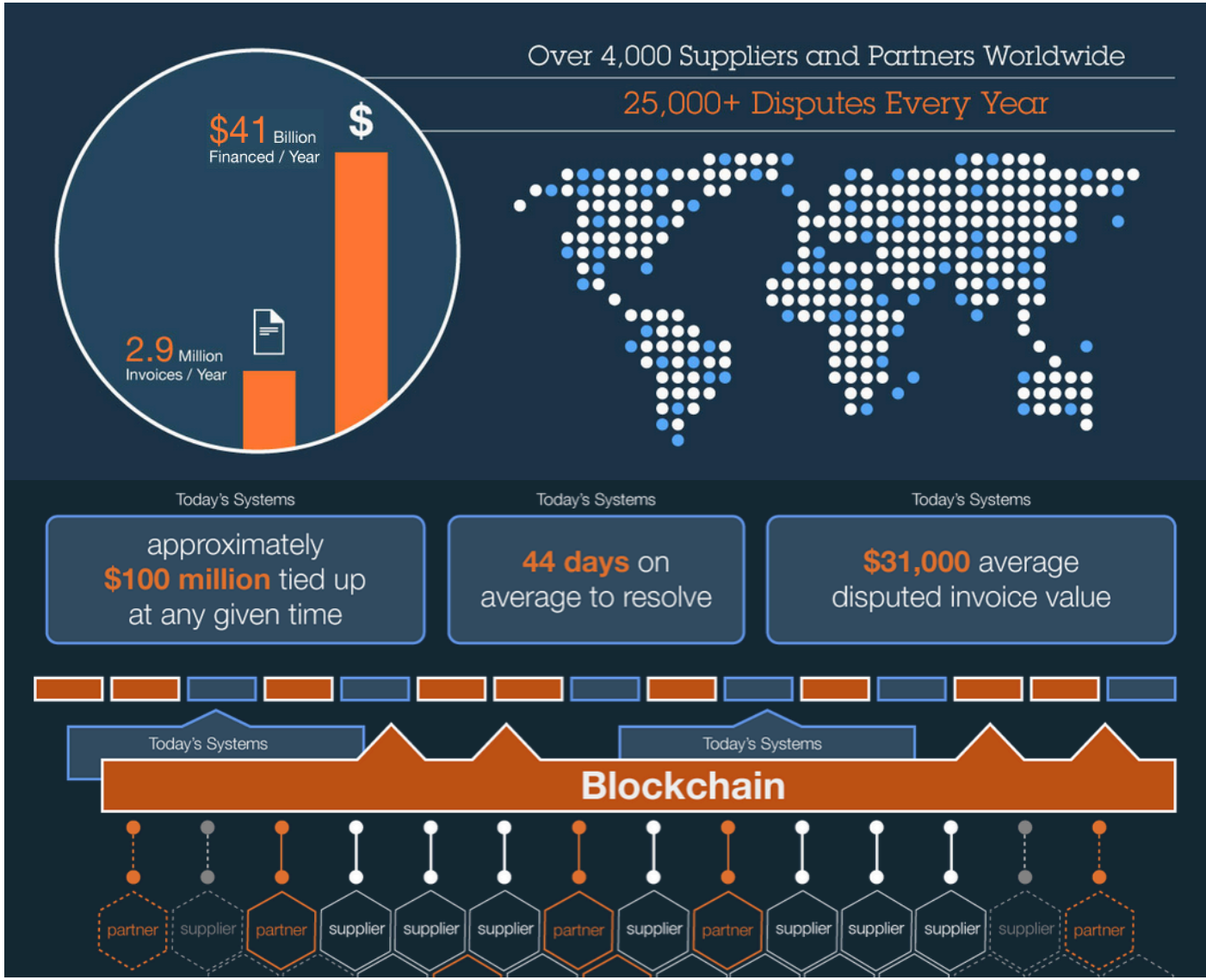
- IBM Global Finance provides a $41bn channel financing per year. There are a number of disputes that take time to resolve and can lock up transactions costing time and money

How?

- **Blockchain provides visibility and provenance end-to-end across supply chain**

Benefits

1. Reduced dispute resolution time by 75%
2. Released working capital from $100m
3. Combine IGF and Supplier info to further expand benefits further
4. In production since Sept 2016

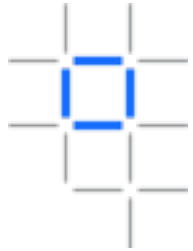


Some key IBM’ers in Blockchain development

Chris Ferris

Christian Cachin

IBM **Blockchain** **Arnaud Le Hors**



What and why we are using: Hyperledger

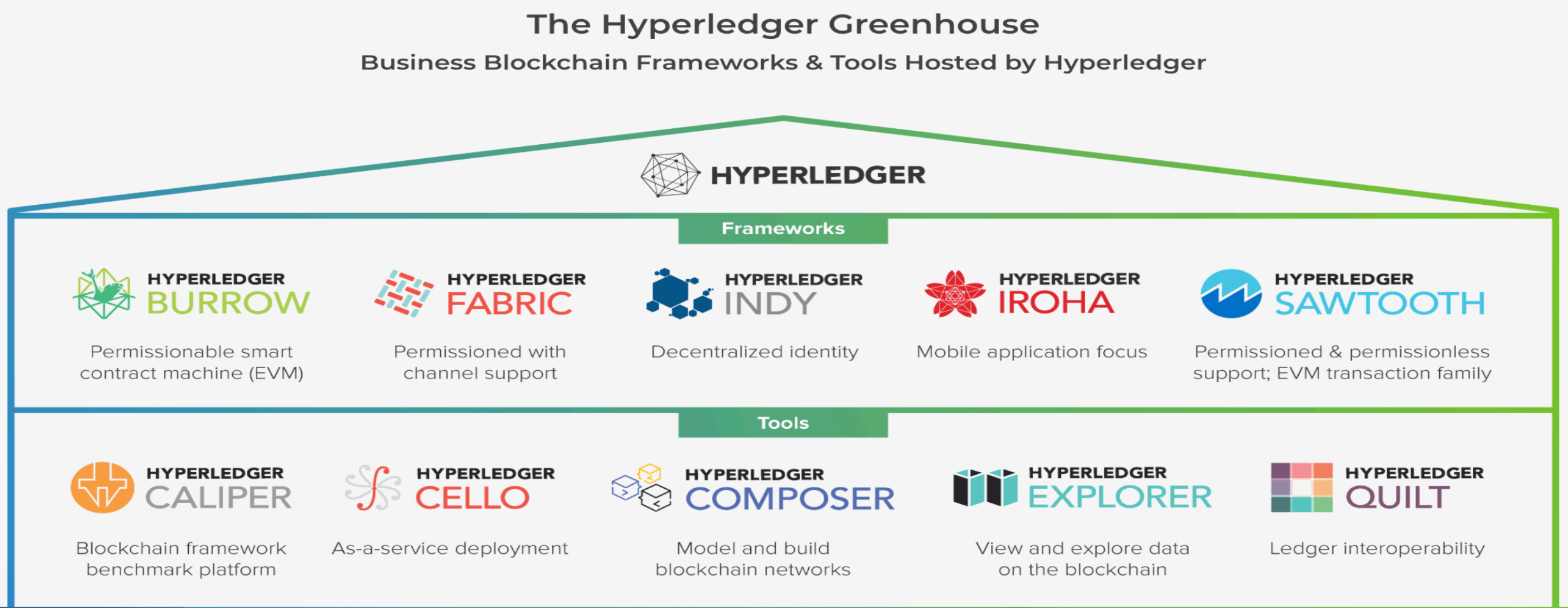
Introducing Hyperledger

Open source collaborative effort to advance **cross-industry blockchain technologies**

Hosted by **The Linux Foundation**, fastest-growing project in LF history

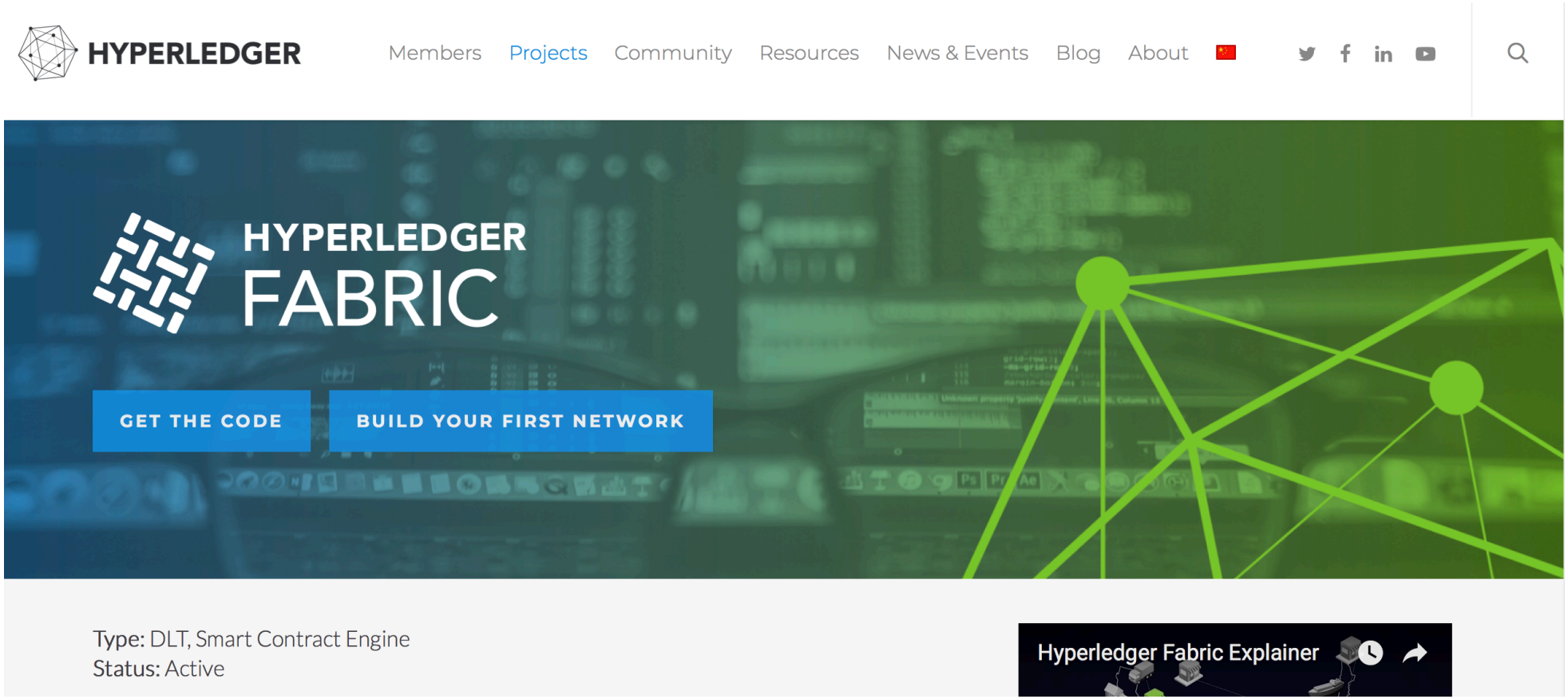
Global collaboration spanning **finance, banking, IoT, supply chain, healthcare, manufacturing, govt, technology** and more.

Which Blockchain do we want to use?



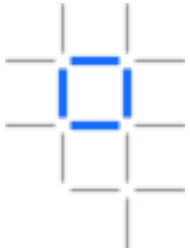
https://www.hyperledger.org/

Hyperledger Fabric



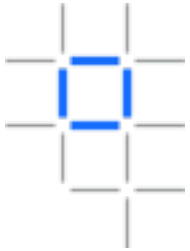
https://www.hyperledger.org/projects/fabric

Languages in Hyperledger Fabric

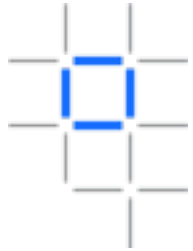


Hyperledger Fabric is written in GoLang. And

Chaincode is written in Go, node.js, and eventually in other programming languages such as Java, that implements a prescribed interface.



Step 1 Architecture



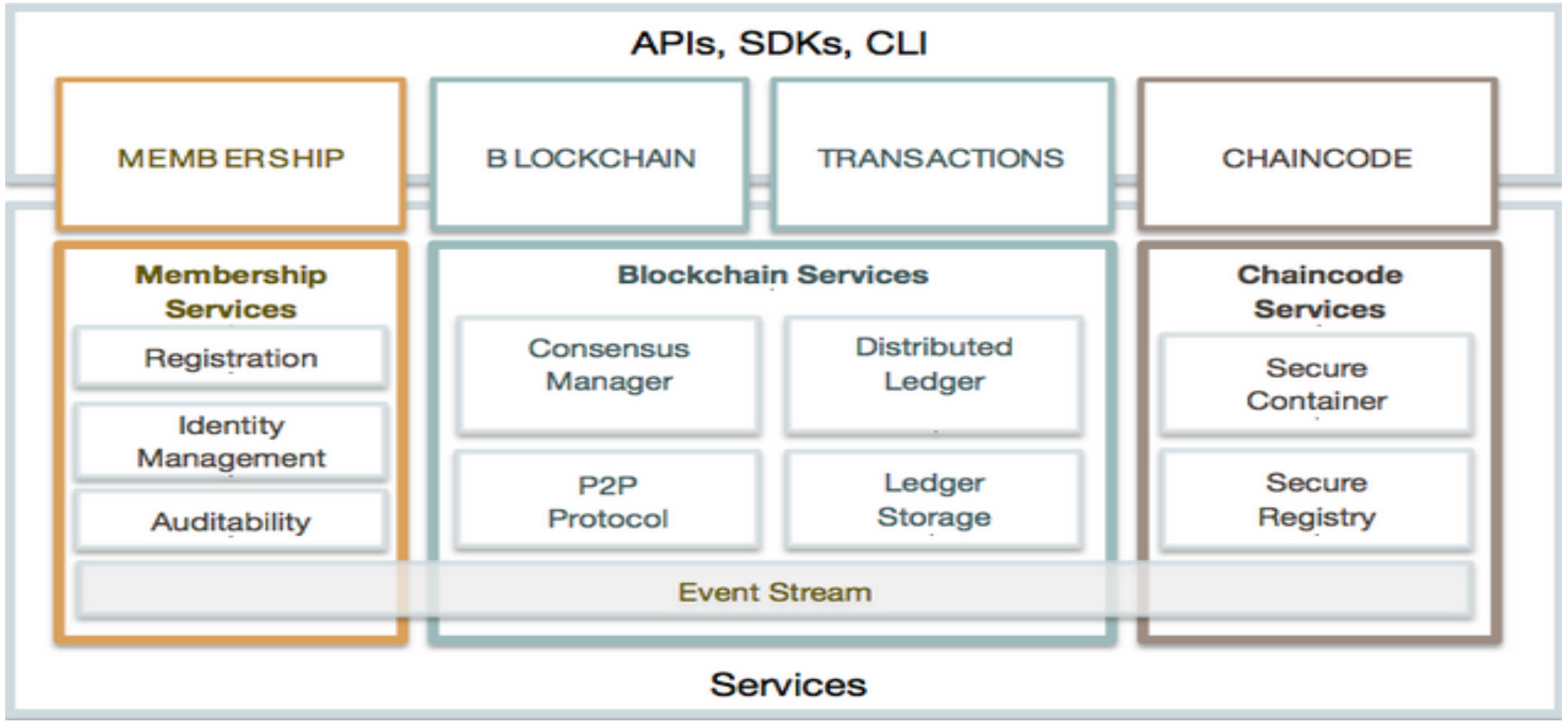
Hyperledger Architecture, Volume 1

Introduction to Hyperledger Business Blockchain Design Philosophy and **Consensus**

This is the first in a series of papers from the Hyperledger Architecture Working Group (WG). These papers describe a generalized reference architecture for permissioned blockchain networks and share the recommendations of the Hyperledger Architecture WG with the end goal of guiding all Hyperledger projects towards modular designs. These papers also serve as a vendor-neutral resource for technical blockchain users and developers interested in using permissioned blockchain networks.

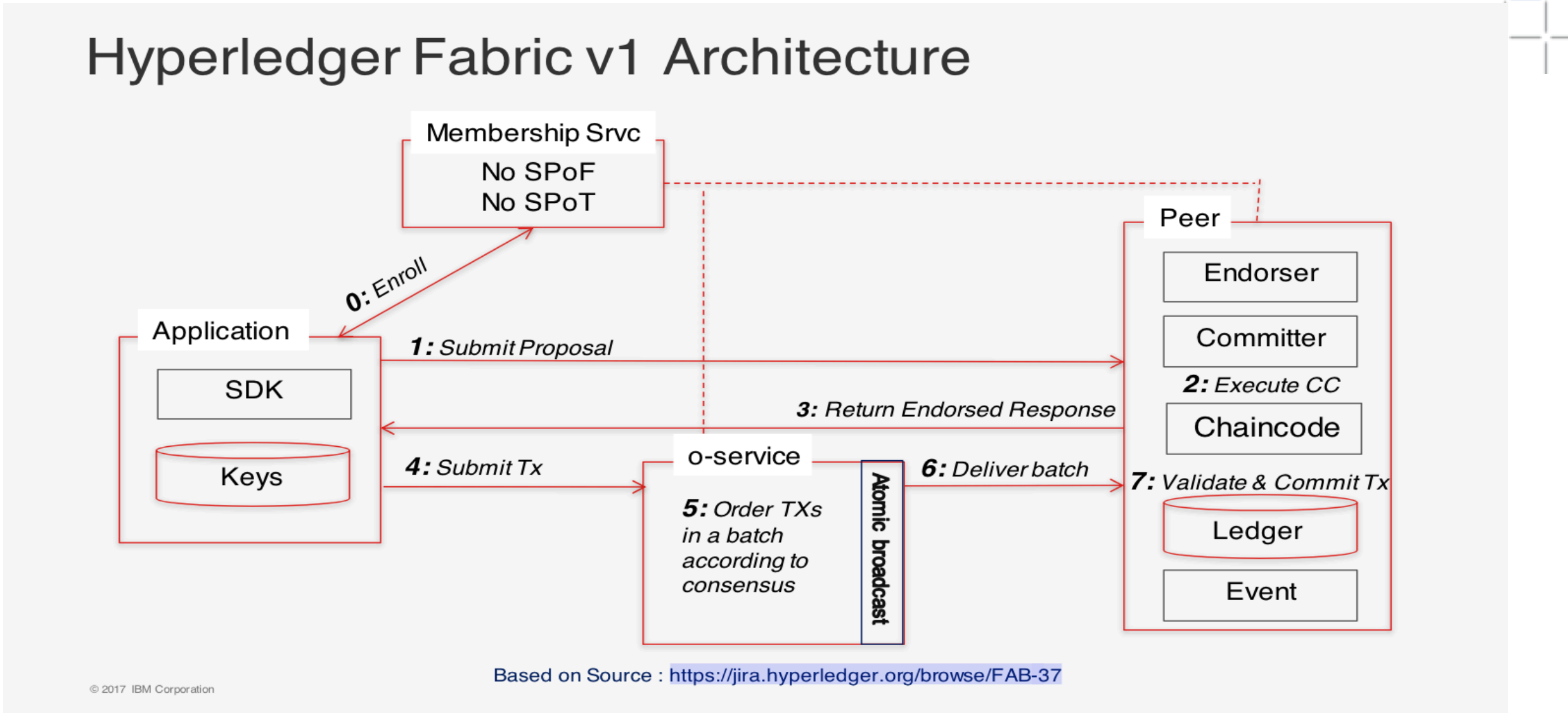
https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

Hyperledger Fabric overview

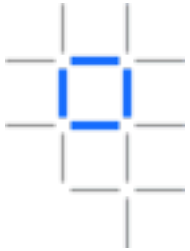


- Hyperledger Fabric allows components, such as consensus and membership services, to be **plug-and-play**.
- Hyperledger Fabric leverages **container technology** (Docker) to host **smart contracts called "chaincode"** that comprise the application logic of the system.
- Chaincode is like Stored Procedures in the traditional database world

Hyperledger Architecture



Hyperledger Fabric has three components



1. **Membership Services** – which manages digital certificates and access,
2. **Ordering service** – which keeps the peers in alignment,
3. and **peers** – which hold **the ledger**.
    1. A peer is also where **chaincode**, also known as smart contracts, are executed.

    Membership and ordering services align well to being centralized.

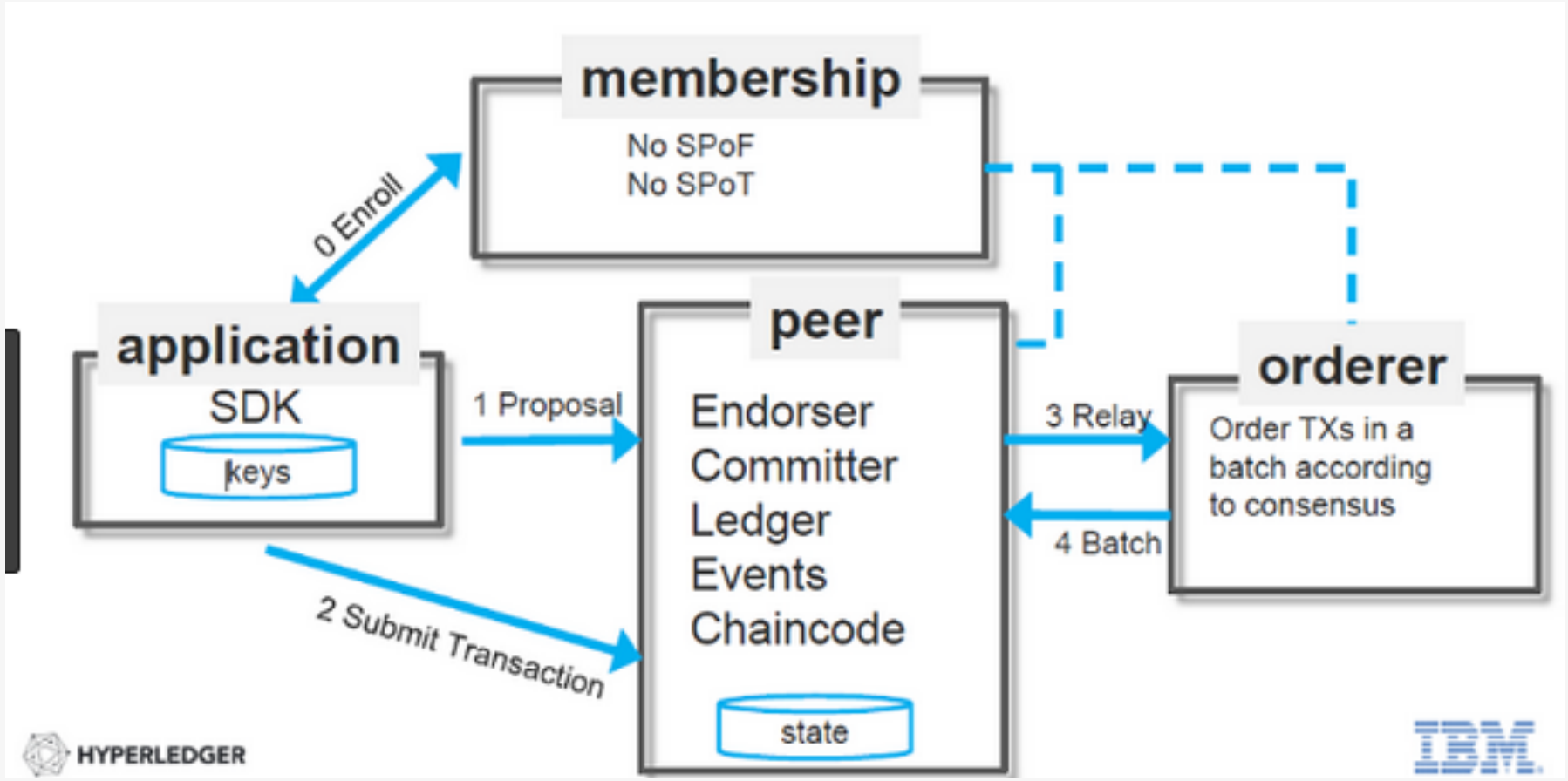
Peers, on the other hand, can be distributed and placed anywhere you desire in the world.

Each member of the Hyperledger network should have their own peer.

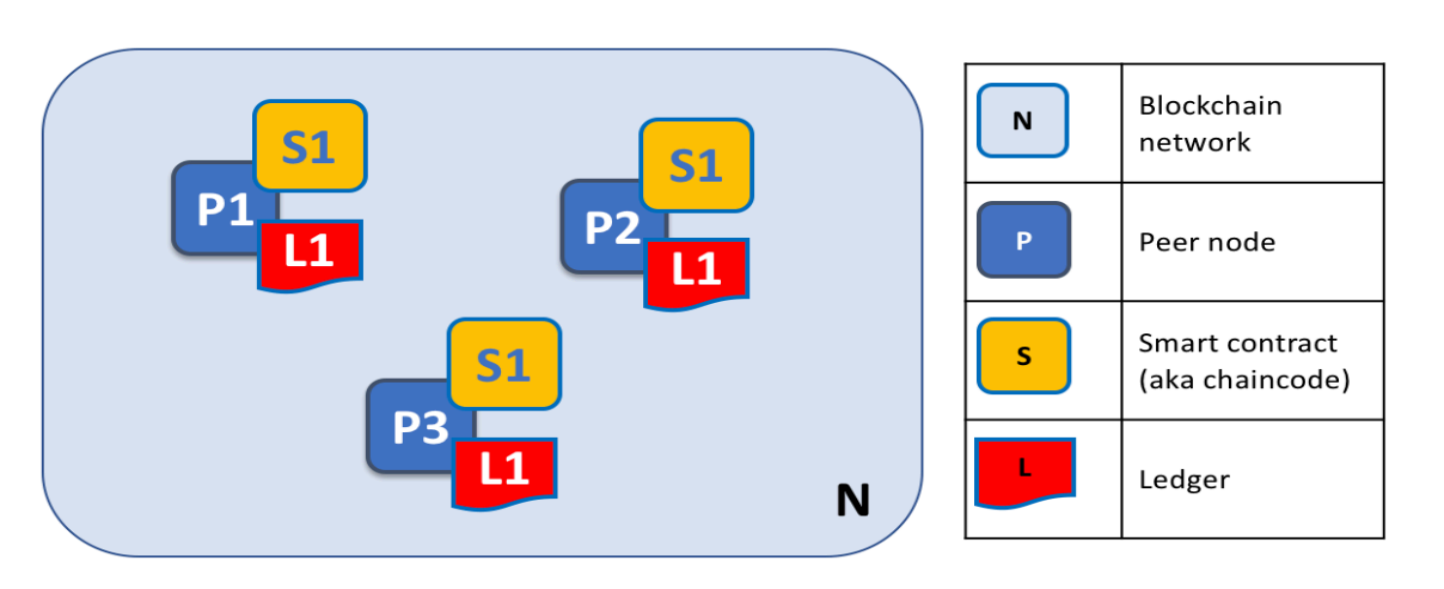
The **channels** (databases) that a peer subscribes to, and transacts against, determines the member's role and capabilities in the blockchain network

https://www.ibm.com/blogs/systems/blockchain-how-should-you-organize-your-peers/

Hyperledger architecture



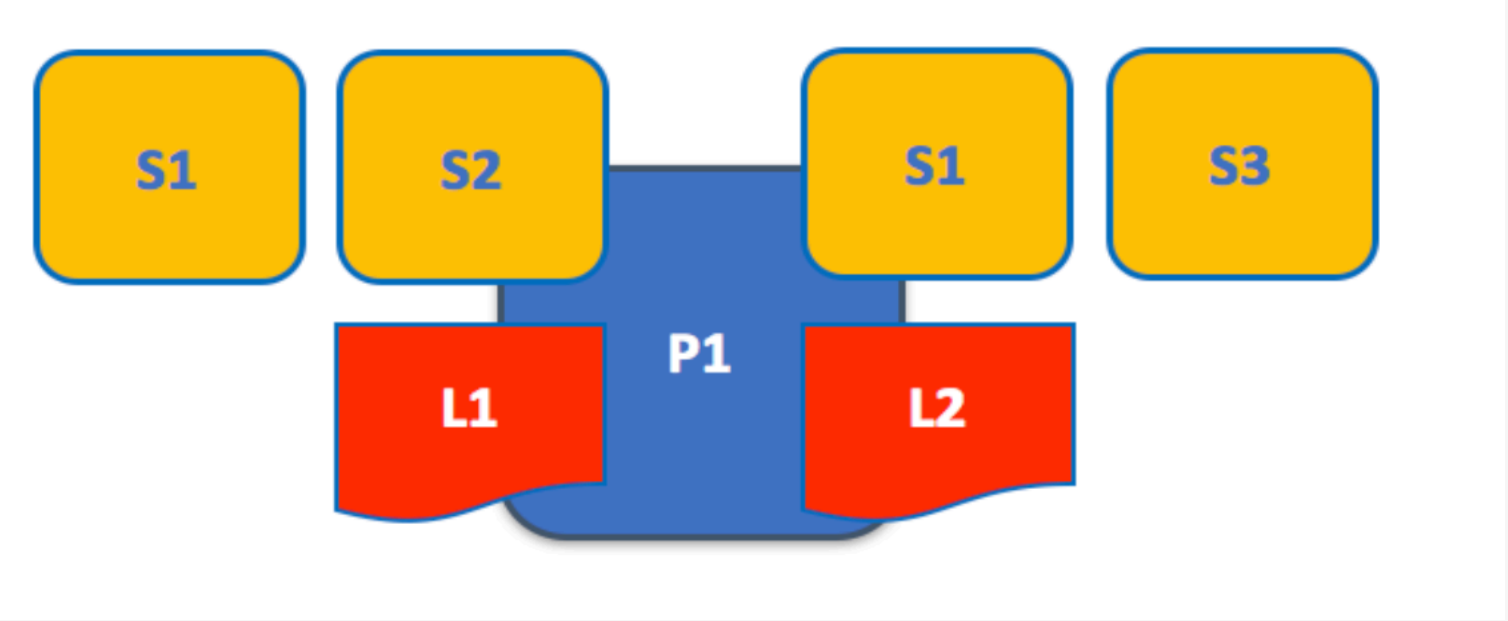
Hyperledger Fabric components



A blockchain network is comprised of peer nodes, each of which can hold copies of ledgers and copies of smart contracts. In this example, the network N consists of peers P1, P2 and P3, each of which maintain their own instance of the distributed ledger L1. P1, P2 and P3 use the same chaincode, S1, to access their copy of that distributed ledger.

https://hyperledger-fabric.readthedocs.io/en/release-1.2/peers/peers.html

Hyperledger Fabric components



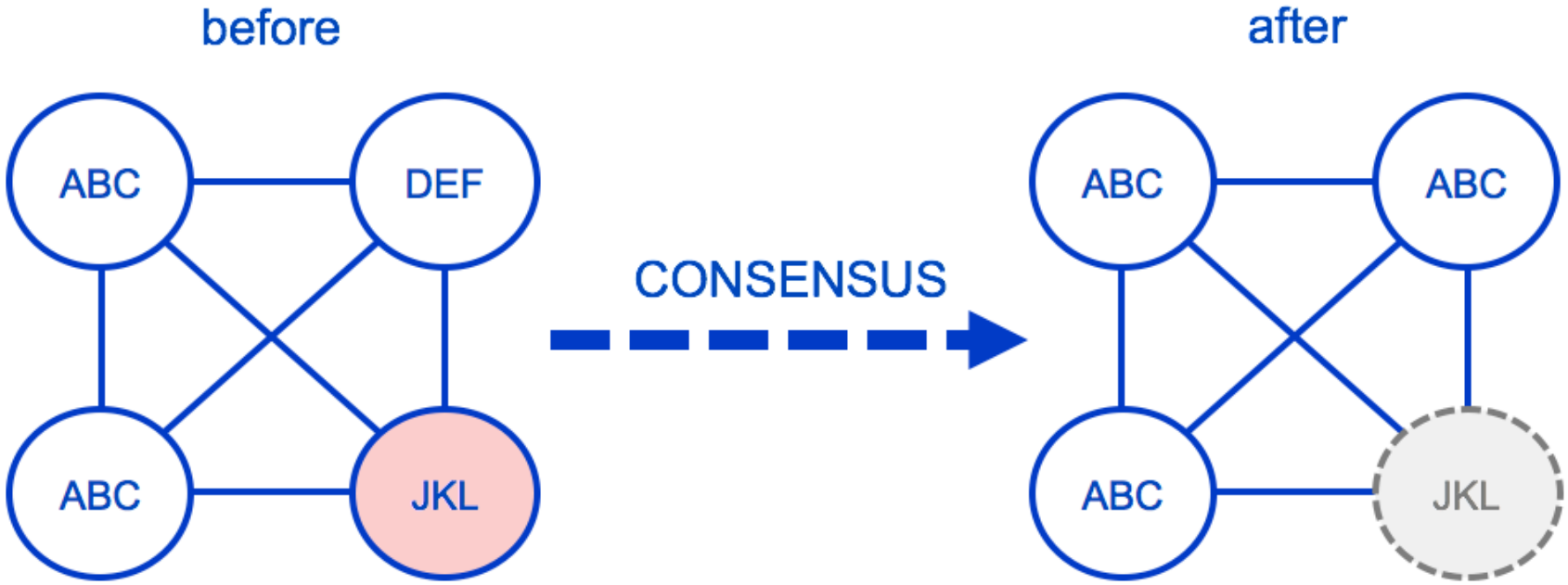
An example of a peer hosting multiple chaincodes. Each ledger can have many chaincodes which access it. In this example, we can see that peer P1 hosts ledgers L1 and L2, where L1 is accessed by chaincodes S1 and S2, and L2 is accessed by S1 and S3. We can see that S1 can access both L1 and L2.

https://hyperledger-fabric.readthedocs.io/en/release-1.2/peers/peers.html

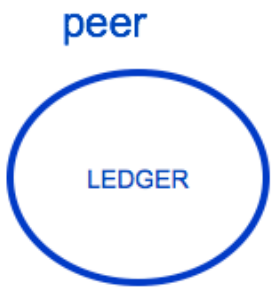
Much more information on Peers and Hyperledger message flow.

https://hyperledger-fabric.readthedocs.io/en/release-1.2/peers/peers.html

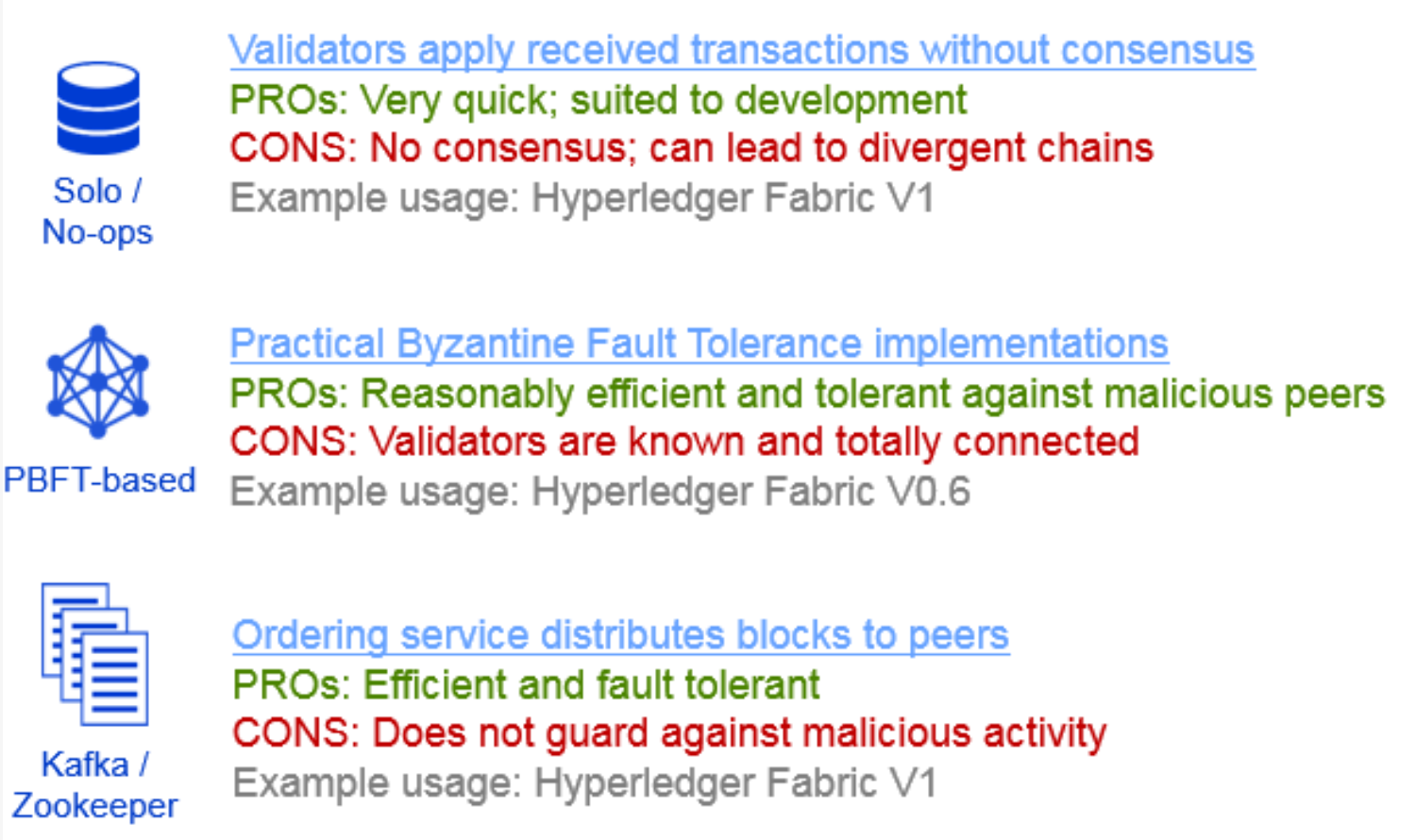
Consensus: The process of maintaining a consistent ledger



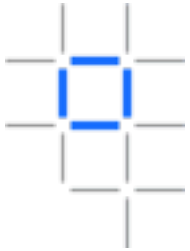
Keep all peers up-to-date
Fix any peers in error
Ignoring all malicious nodes



Consensus algorithms have different strengths and weaknesses



Hyperledger Fabric Architecture

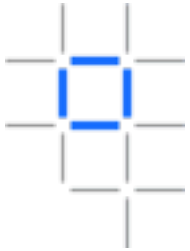


Architecturally, the fabric is extremely flexible, but in practice consider two recommendations.

- **First, you only need one or two peers**. Per a given channel, peers are identical. They execute the same transactions and receive the same code updates in the exact same order as all other peers on that channel. **There is no reason to have lots of peers**.
  - **Get two peers if you need the availability** – which I would recommend.
- **Secondly, the front-end systems that hold the Hyperledger SDKs (those systems that provide RESTful interfaces) should be aligned to a member's peers.**
  - Which means you want your front-end systems talking to your peers and you want other member's front-end systems talking to their peers.
  - **The idea of an SDK talking to all of the peers for consensus purposes makes little to no sense when they will all return the same results.**
  - **One SDK only needs to talk to one peer or two, for high availability,** to get the optimal performance and functional capabilities of the fabric.

https://www.ibm.com/blogs/systems/blockchain-how-should-you-organize-your-peers/

Which user roles do you need in a business network?



- **The consortium owner**: **This is the organization** with the vision for the network, its business benefits and how it will disrupt how business is done. Today, for practical purposes, it is best if there is only one consortium owner, since that entity will be responsible for making the network work.
- **Members:** **They are part of the Hyperledger network**. They have a peer, which means they can store data, execute chain code (smart contracts), and participate in the consensus process (if the consortium owner allows it). Members bring substance and validity to the network, and it is their data and intelligence that is shared. As the network matures, members can move up in status to become consortium owners.
- **General users:** **These are users of the network** and they are sponsored and enabled by the consortium owner and the members. **They can access and input data, but they do not own data**. As with members, as the network matures, general users can become members if they bring value to the network.

https://www.ibm.com/blogs/systems/blockchain-who-should-be-in-your-consortium/

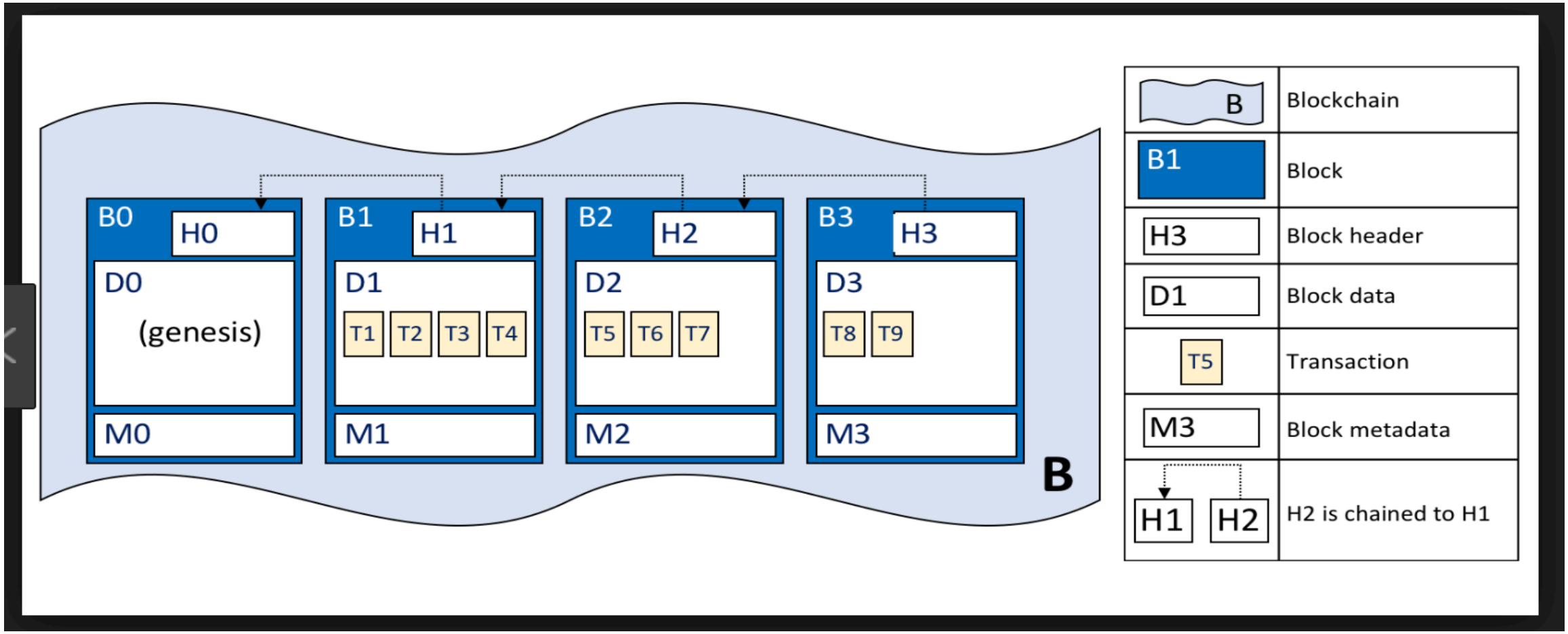
Hyperledger Fabric Performance

The performance numbers obtained show that Hyperledger Fabric deployed in a single cloud data center achieves an end-to-end throughput

3,500 transactions per second with latency less than one second.

https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/

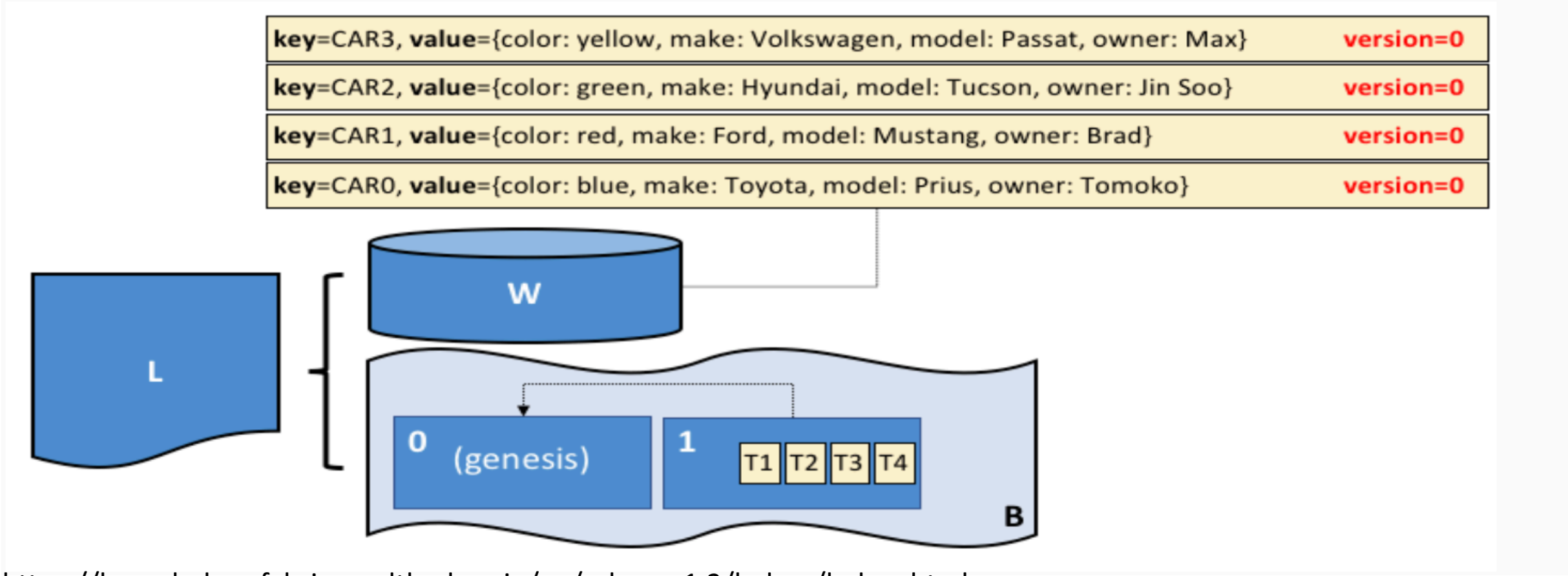
The chain of transaction blocks in the Blockchain



Each block is made up of a number of transactions, the result of chain code invocations

IBM **Blockchain** **https://hyperledger-fabric.readthedocs.io/en/release-1.2/ledger/ledger.html**

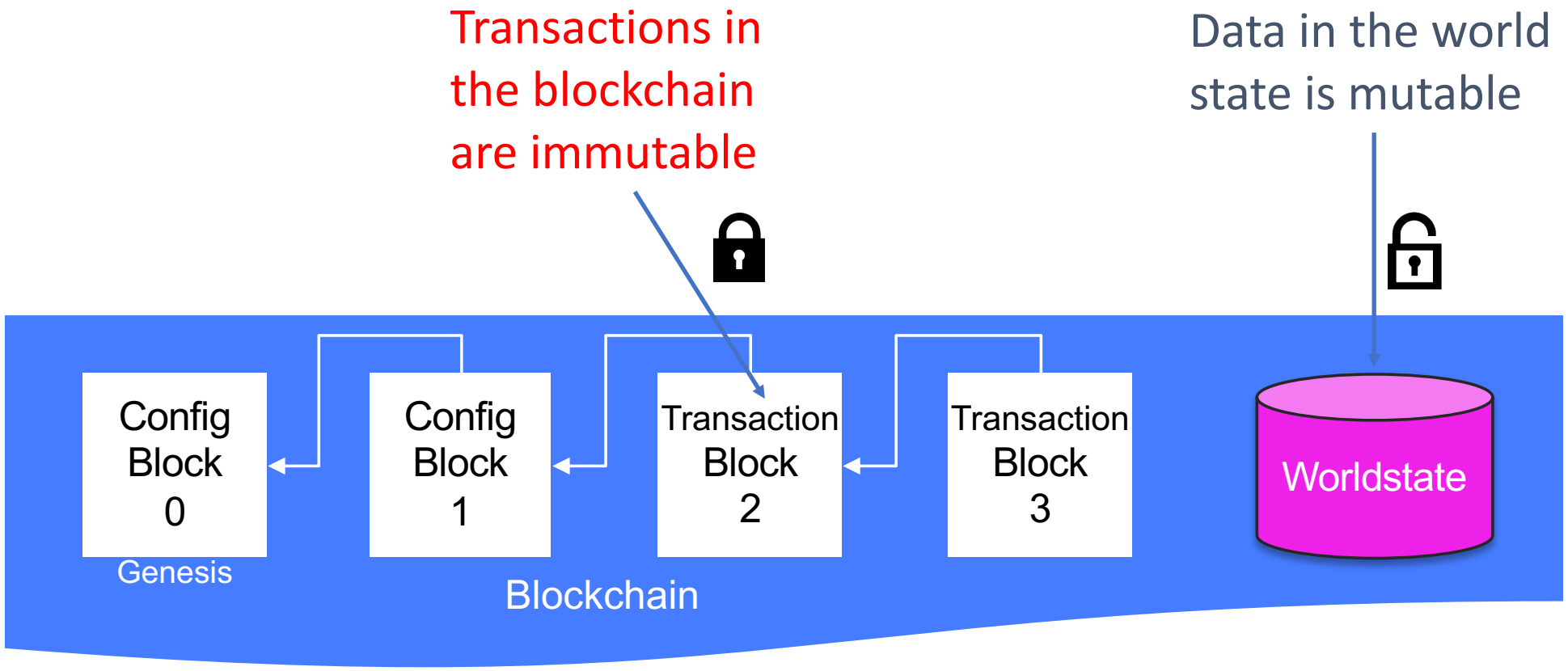
Transactions on the Blockchain from chain code invocations



https://hyperledger-fabric.readthedocs.io/en/release-1.2/ledger/ledger.html

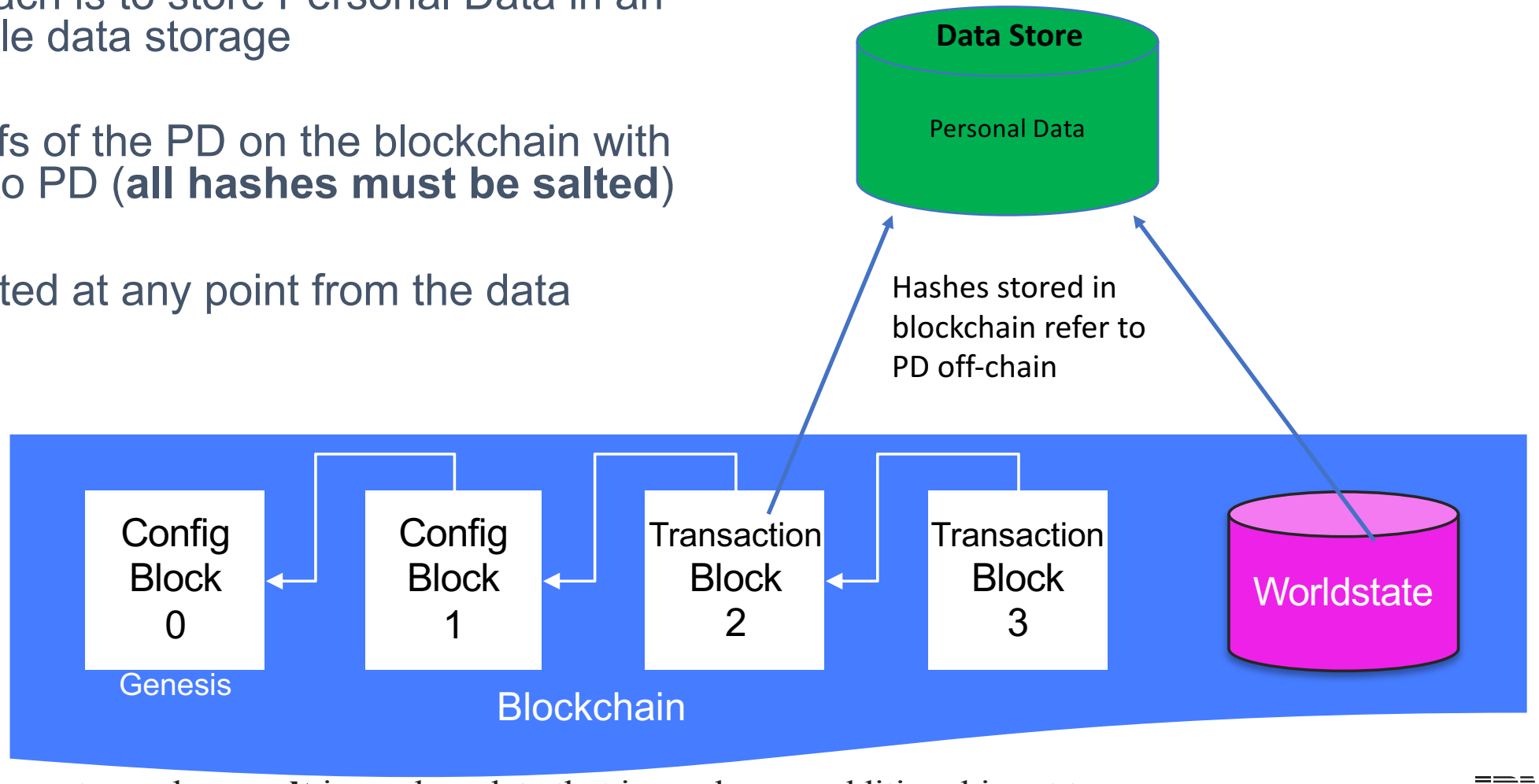
The “Right to Erasure” and blockchain

Enables an individual to request the deletion or removal of their personal data



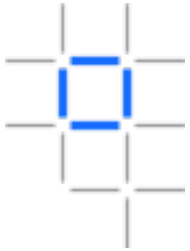
General Data Protection Regulation (GDPR) Solution – Store data off-chain

- **What**
  - The only approach is to store Personal Data in an off-chain mutable data storage
- **How**
  - Store only proofs of the PD on the blockchain with hashes linking to PD (**all hashes must be salted**)
- **Why**
  - PD can be deleted at any point from the data store(s)



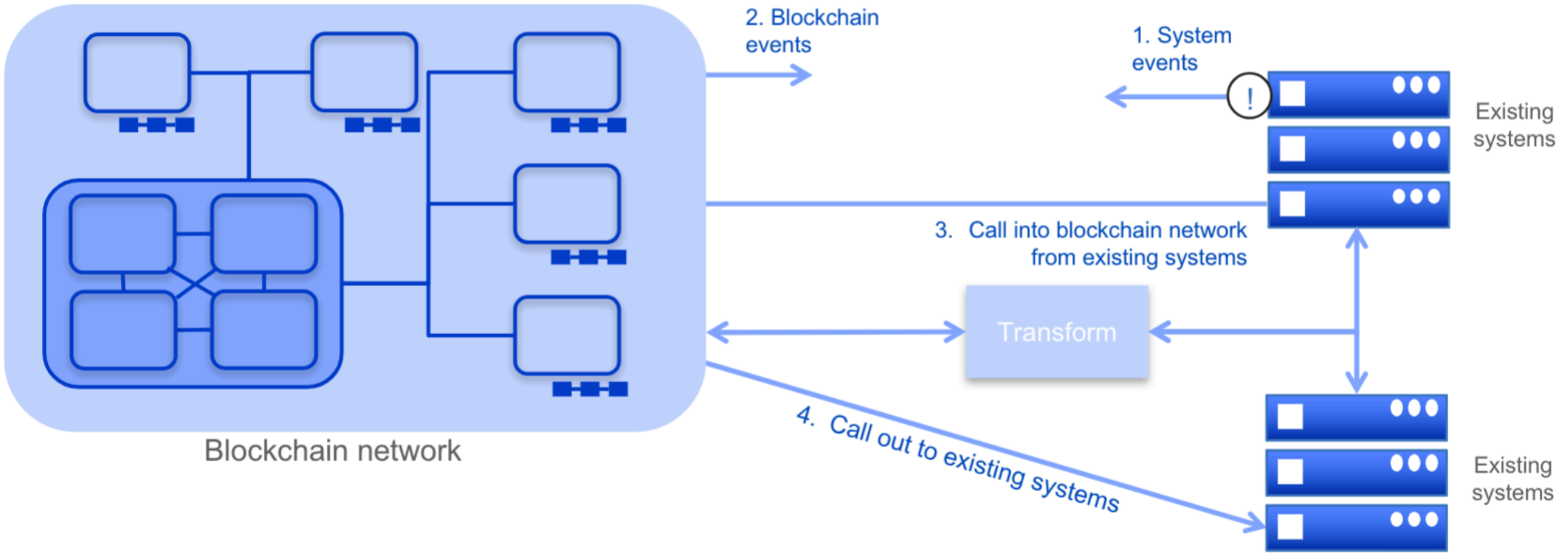
In cryptography, **a salt** is random data that is used as an additional input to a one-way function that "**hashes**" data, a password or passphrase

Off chain, the problem with the Systems of Record



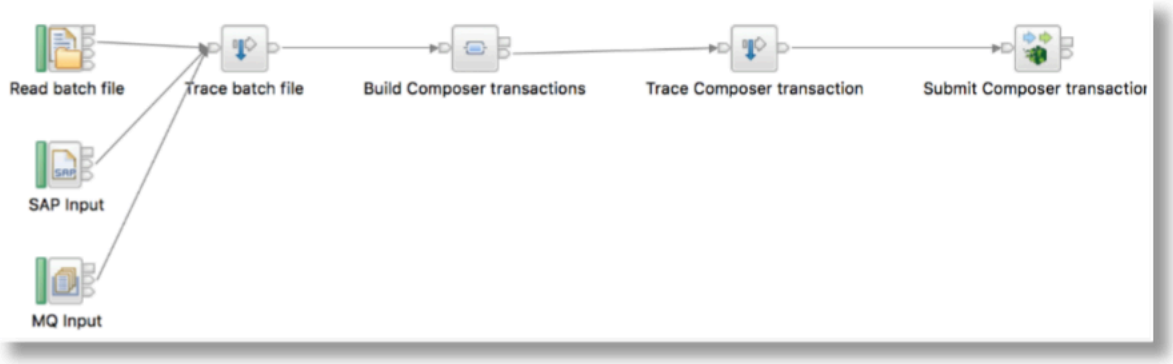
- All data can not be stored on the Blockchain
  - Example X-ray images
  - Diamonds
  - Art works
  - Automobiles
  - Aircraft parts
  - Food

Integrating with existing systems - possibilities



Integrating with existing systems – using middleware

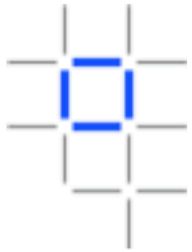
- Blockchain is a network system of record
- Two-way exchange
  - Events from blockchain network create actions in existing systems
  - Cumulative actions in existing systems result in Blockchain interaction



- Transformation between blockchain and existing systems' formats
  - GBO, ASBO is most likely approach
  - Standard approach will be for gateway products to bridge these formats
  - Gateway connects to peer in blockchain network and existing systems

- Smart contracts can call out to existing systems
  - Query is most likely interaction for smart decisions
    - e.g. all payments made before asset transfer?
    - Warning: Take care over predictability… transaction must provide same outputs each time it executes

Non-determinism in blockchain



- Blockchain is a distributed processing system
  - Smart contracts are run multiple times and in multiple places
  - As we will see, smart contracts need to run deterministically in order for consensus to work
    - Particularly when updating the world state
- It's particularly difficult to achieve determinism with off-chain processing
  - Implement services that are guaranteed to be consistent for a given transaction, or
  - Detect duplicates for a transaction in the blockchain, middleware or external system

random()

getExchangeRate()

getDateTime()

getTemperature()

incrementValue inExternalSystem(…)

Facets of distributed, shared ledgers

Network nodes both **generate their own data** and **verify data** generated by others	Contain historic record of verified transactions that are **easily auditable**	**Distributed Consensus** eliminates costly and inefficient reconciliation processes
No central repository – each node stores identical copies of the ledger	**Resilient** due to network power and cryptographic integrity	Large economic **disincentive** for **malicious** actors

HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

https://www.youtube.com/watch?v=-O3ouBdG3Xg

Spectrum of Blockchains

Permissionless Public	Permissionless Private	Permissioned Public	Permissioned Private
Bitcoin, Ethereum	Public Polls	Land Titles, University Degrees	Medical Records

Permissioned vs. Permissionless: Who can write to a Blockchain, i.e. accessibility
Public vs Private: who can read from a Blockchain, i.e. visibility

https://www.youtube.com/watch?v=-O3ouBdG3Xg

Why not use a Database?

- A traditional database is **centralized**
- Everyone needs to **trust** the administrator managing the database
- There's typically **no immutability or provenance**

- Distributed databases do not alleviate the **trust** issue
- There are now **more copies** to worry about and **more administrators**

- **Blockchain** allows the concept of a distributed database to be deployed across an **untrusted network**
- Something a traditional database cannot handle

Traditional databases cannot be used in untrusted networks

Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains

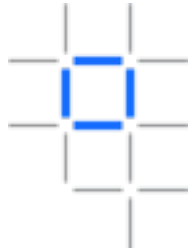
Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, Jason Yellick

Fabric is a modular and extensible open-source system for deploying and operating permissioned blockchains and one of the Hyperledger projects hosted by the Linux Foundation (www.hyperledger.org).
Fabric is the first truly extensible blockchain system for running distributed applications. It supports modular consensus protocols, which allows the system to be tailored to particular use cases and trust models. Fabric is also the first blockchain system that runs distributed applications written in standard, general-purpose programming languages, without systemic dependency on a native cryptocurrency. This stands in sharp contrast to existing blockchain platforms that require "smart-contracts" to be written in domain-specific ….

This paper describes Fabric, its architecture, the rationale behind various design decisions, its most prominent implementation aspects, as well as its distributed application programming model. We further evaluate Fabric by implementing and benchmarking a Bitcoin-inspired digital currency. We show that Fabric achieves end-to-end throughput of more than 3500 transactions per second in certain popular deployment configurations, with sub-second latency, scaling well to over 100 peers.

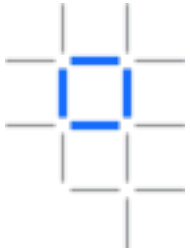
HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

https://arxiv.org/abs/1801.10228



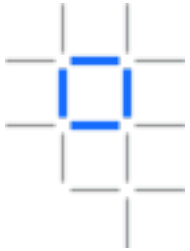
Step 2 Use Cases

What makes a good Blockchain Use Case?



- A Business problem that cannot easily be solved with existing techniques
- An identifiable business network
  - With a shared ledger
  - With Participants, Assets and Transactions
- A need for trust
  - Consensus, Immutability, Finality and Provenance

Which user roles do you need in a business network?



- **The consortium owner**: **This is the organization** with the vision for the network, its business benefits and how it will disrupt how business is done. Today, for practical purposes, it is best if there is only one consortium owner, since that entity will be responsible for making the network work.
- **Members:** **They are part of the Hyperledger network**. They have a peer, which means they can store data, execute chain code (smart contracts), and participate in the consensus process (if the consortium owner allows it). Members bring substance and validity to the network, and it is their data and intelligence that is shared. As the network matures, members can move up in status to become consortium owners.
- **General users:** **These are users of the network** and they are sponsored and enabled by the consortium owner and the members. **They can access and input data, but they do not own data**. As with members, as the network matures, general users can become members if they bring value to the network.

https://www.ibm.com/blogs/systems/blockchain-who-should-be-in-your-consortium/

Blockchain Use Cases

Supply chain, asset registration, identity services, fraud prevention and compliance

IBM Blockchain government point of view

PDF See the infographic (50.6KB)

Payment and digital currency

IBM's universal blockchain payments solution

PDF See the infographic (1.4MB)

Payment and digital currency

IBM announces major blockchain solution to speed up global payments

→ Read the press release

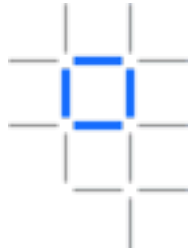
European trade finance network: we.trade

Spreading opportunity to small and medium sized businesses in traditionally underserved markets.

Watch the video (02:48)

https://www.ibm.com/blockchain/use-cases/

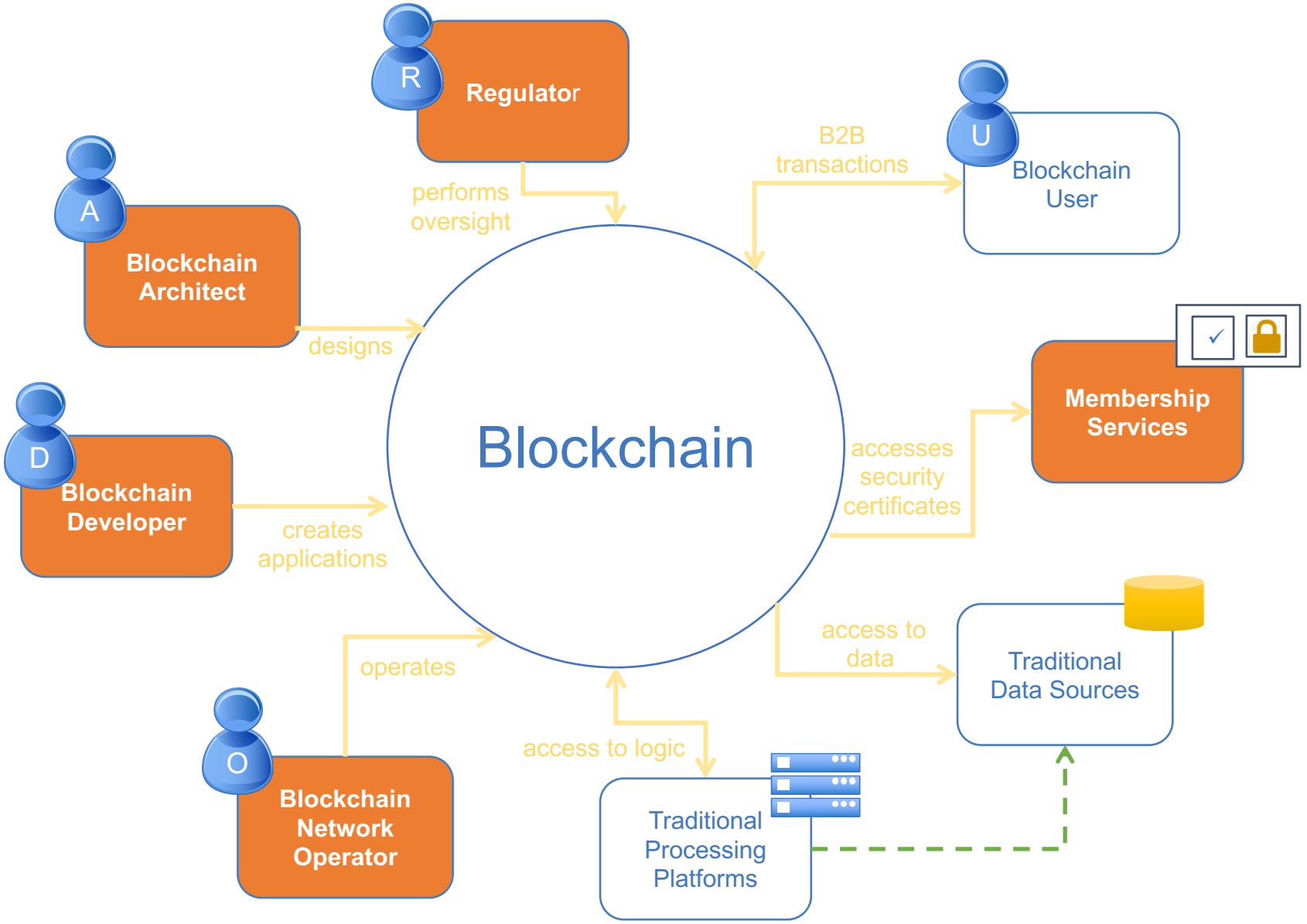
What makes a good Blockchain Use Case?

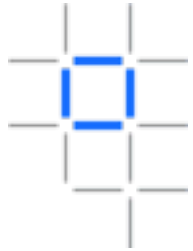


- Identifying a good blockchain use-case is not always easy!
  - However there should always be:

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

Actors in a Blockchain solution





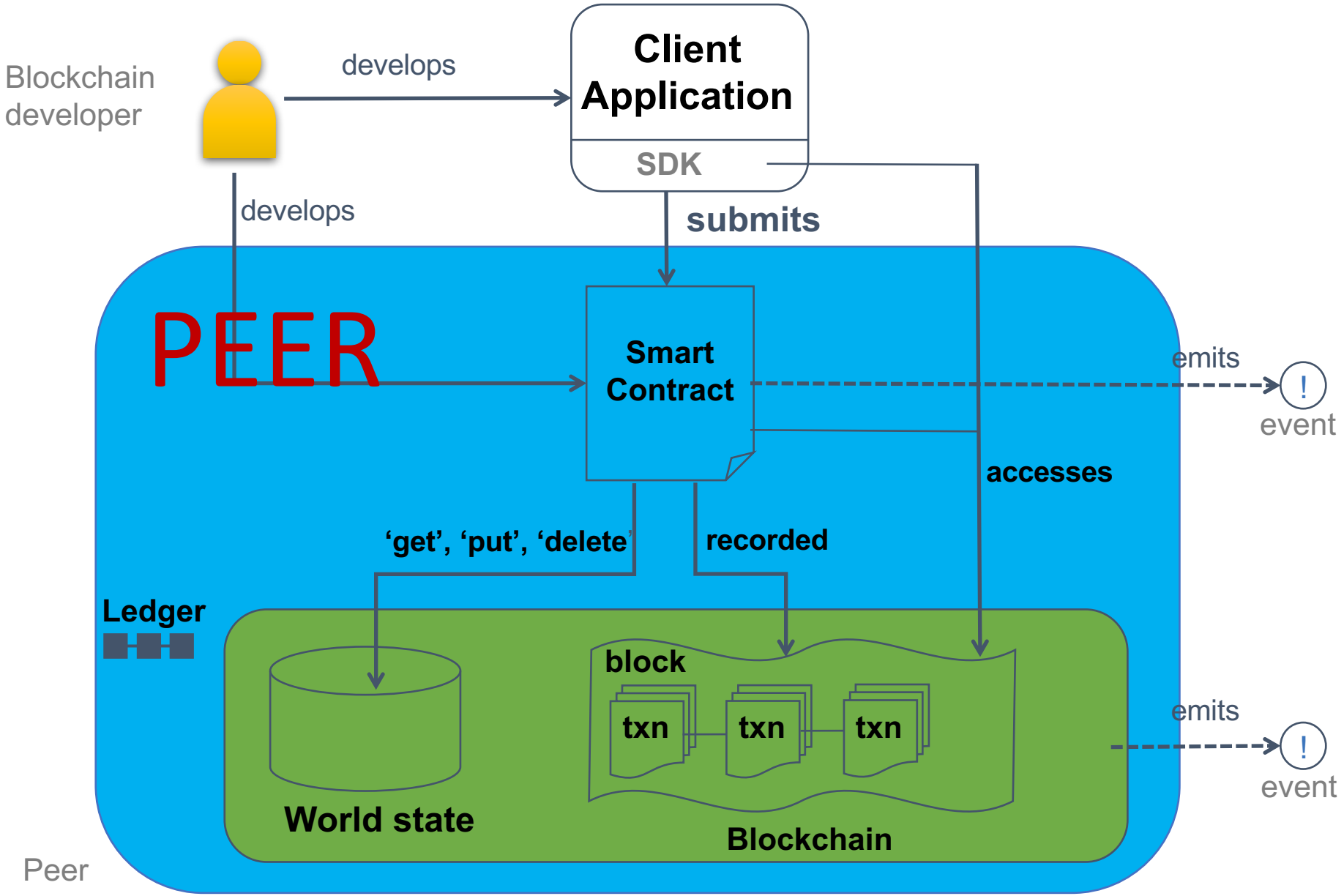
Step 3: Programming the ledger with Chaincode alias Smart Contracts

Chaincode and Smart Contracts

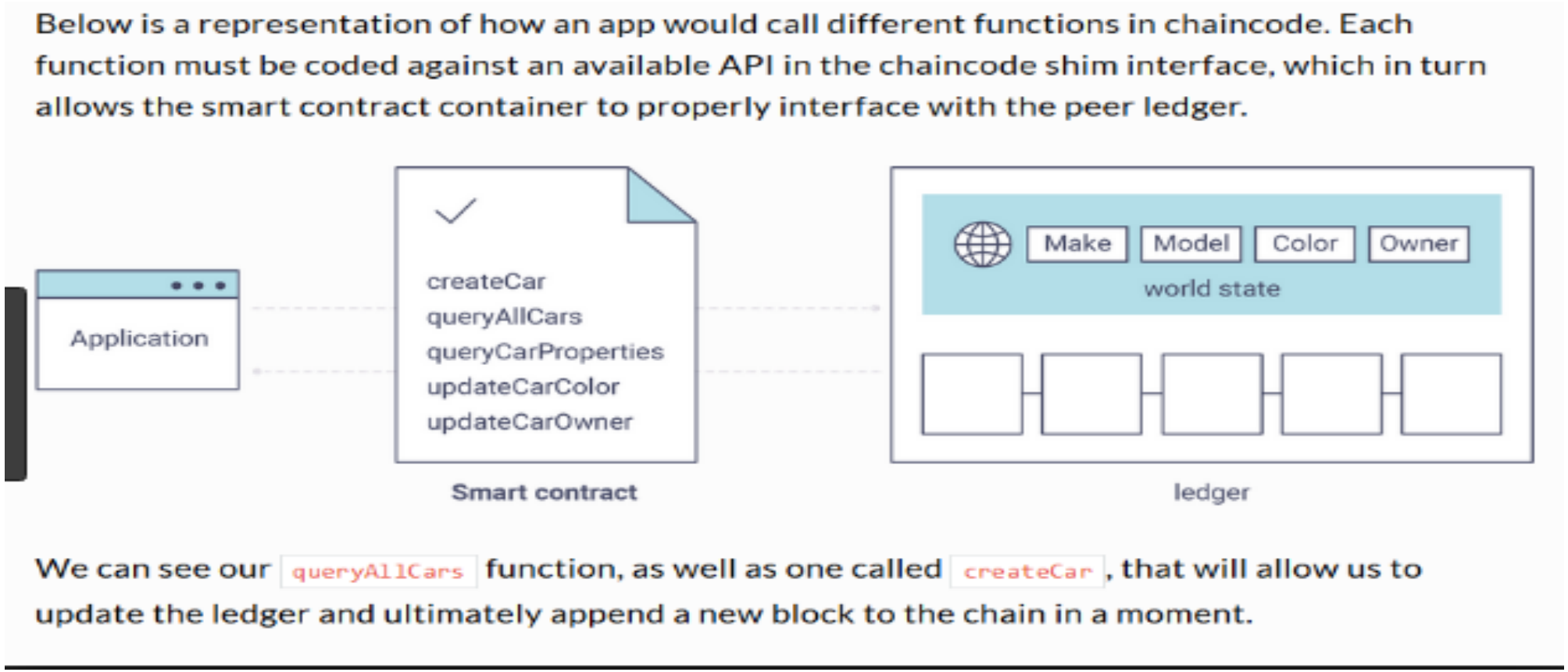
- **Chaincode is a program, written in Go, node.js, and eventually in other programming languages such as Java, that implements a prescribed interface. Chaincode runs in a secured Docker container isolated from the endorsing peer process.**
- **Similar to smart Contracts and Stored Procedures**

© 2017 IBM Corporation

Programming the ledger



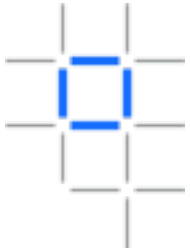
Chaincode, aka Smart Contracts



https://hyperledger-fabric.readthedocs.io/en/release-1.2/chaincode.html

Like Stored Procedures in databases

Two Chaincode Personas

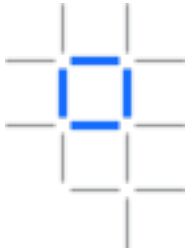


- One, from the perspective of an application developer developing a blockchain application/solution entitled **Chaincode for Developers**, and the other,

- **Chaincode for Operators** oriented to the blockchain network operator who is responsible for managing a blockchain network/

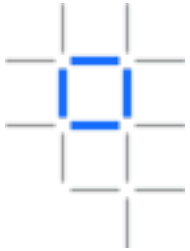
https://hyperledger-fabric.readthedocs.io/en/release-1.2/chaincode.html

More on Chaincode



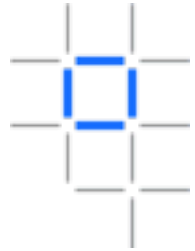
- A chaincode typically handles business logic agreed to by members of the network, so it may be considered as a “smart contract”.
- State created by a chaincode is scoped exclusively to that chaincode and can’t be accessed directly by another chaincode.
- However, within the same network, given the appropriate permission a chaincode may invoke another chaincode to access its state.

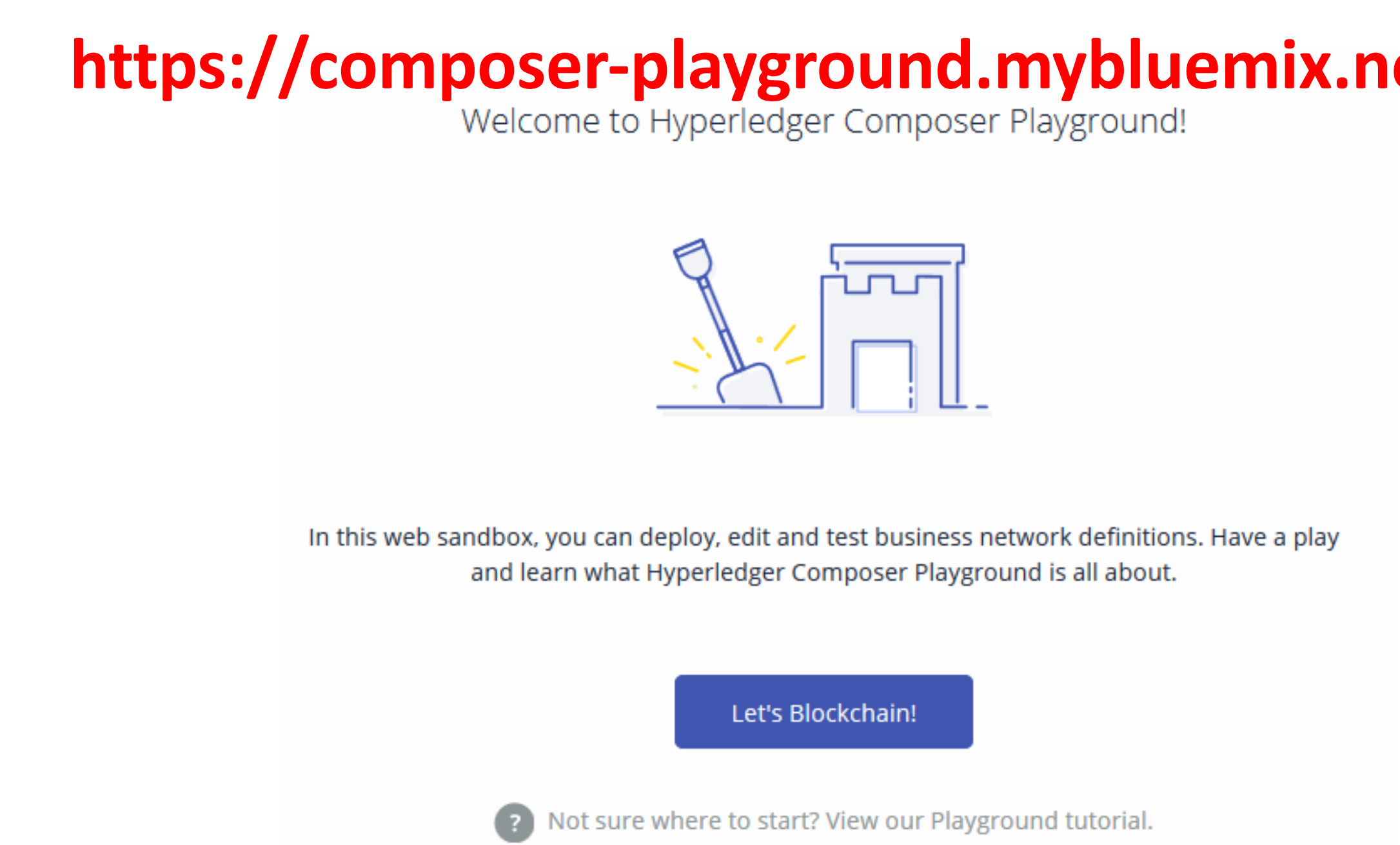
https://hyperledger-fabric.readthedocs.io/en/release-1.2/chaincode.html



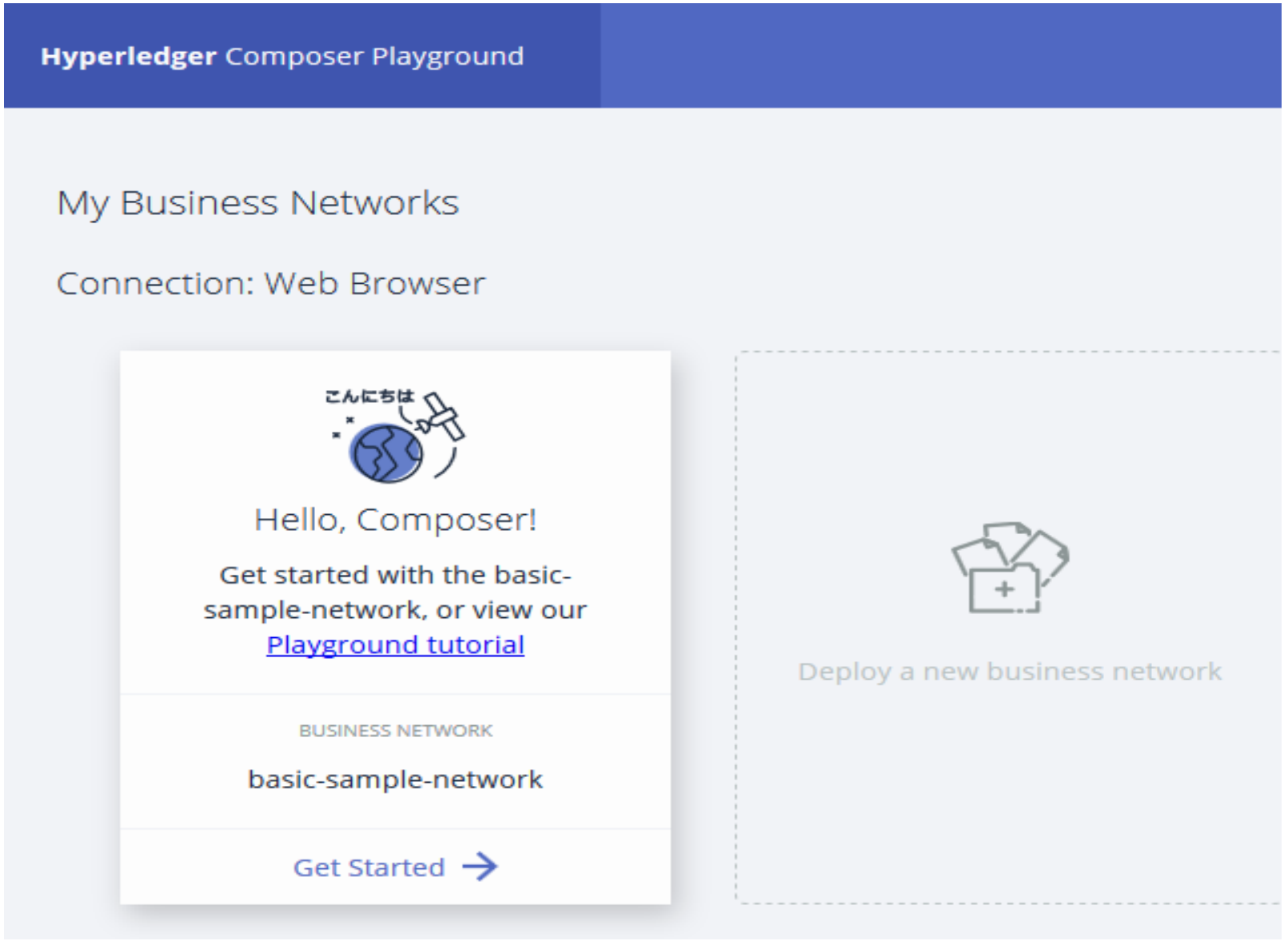
Step 3.1: Programming with Hyperledger Composer

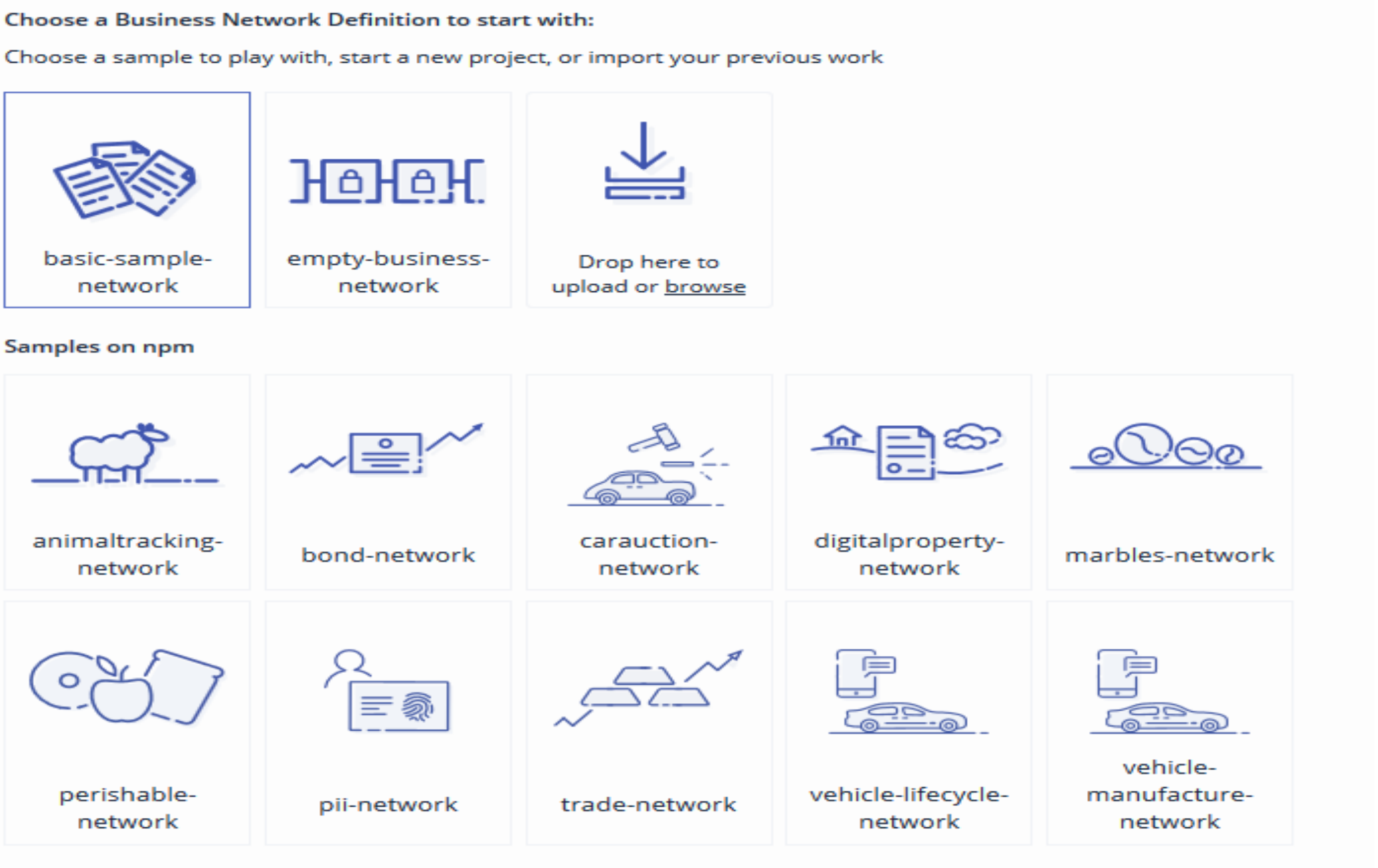
https://composer-playground.mybluemix.net/

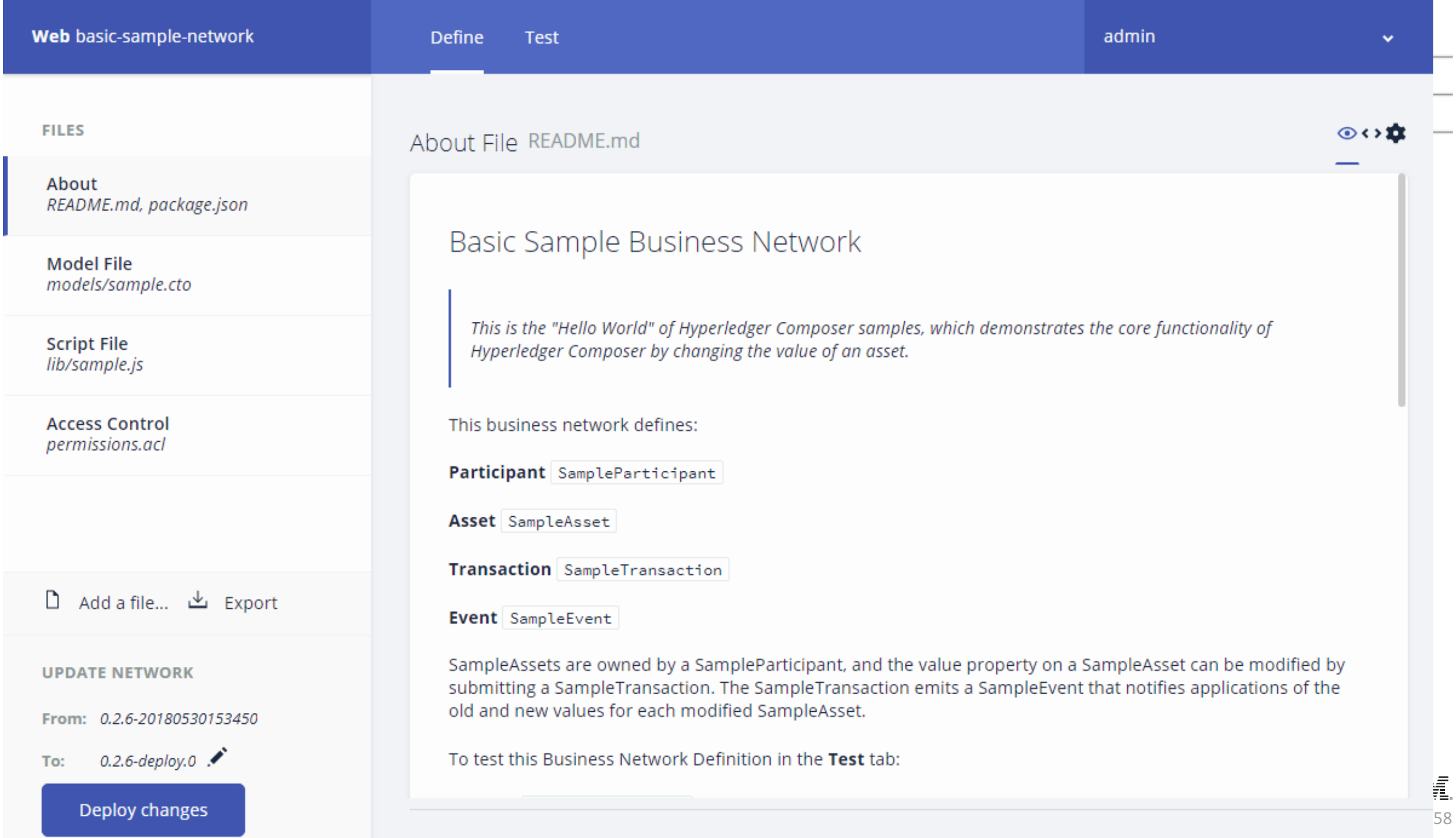




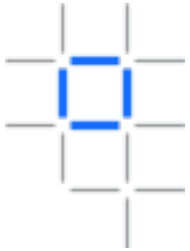
https://hyperledger.github.io/composer/latest/tutorials/playground-tutorial


Hyperledger Composer Playground
My Business Networks
Connection: Web Browser
こんにちは
Hello, Composer!
Get started with the basic-sample-network, or view our Playground tutorial
BUSINESS NETWORK
basic-sample-network
Get Started
Deploy a new business network


Choose a Business Network Definition to start with:
Choose a sample to play with, start a new project, or import your previous work
basic-sample-network
empty-business-network
Drop here to upload or browse
Samples on npm
animaltracking-network
bond-network
carauction-network
digitalproperty-network
marbles-network
perishable-network
pii-network
trade-network
vehicle-lifecycle-network
vehicle-manufacture-network


Web basic-sample-network
Define
Test
admin
FILES
About
README.md, package.json
Model File
models/sample.cto
Script File
lib/sample.js
Access Control
permissions.acl
Add a file...
Export
UPDATE NETWORK
From: 0.2.6-20180530153450
To: 0.2.6-deploy.0
Deploy changes
About File README.md
Basic Sample Business Network
This is the "Hello World" of Hyperledger Composer samples, which demonstrates the core functionality of Hyperledger Composer by changing the value of an asset.
This business network defines:
Participant SampleParticipant
Asset SampleAsset
Transaction SampleTransaction
Event SampleEvent
SampleAssets are owned by a SampleParticipant, and the value property on a SampleAsset can be modified by submitting a SampleTransaction. The SampleTransaction emits a SampleEvent that notifies applications of the old and new values for each modified SampleAsset.
To test this Business Network Definition in the Test tab:

Hyperledger Composer Modeling Language



Hyperledger Composer includes an object-oriented modeling language that is used to define the domain model for a business network definition

1. A single namespace. All resource declarations within the file are implicitly in this namespace.

2. A set of resource definitions, encompassing assets, transactions, participants, and events.

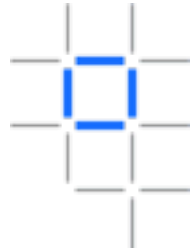
3. Optional import declarations that import resources from other namespaces.

https://hyperledger.github.io/composer/latest/reference/cto_language

Model File models/sample.cto

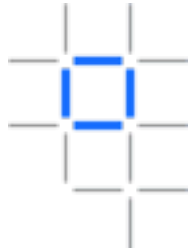
```
14
15  /**
16   * Sample business network definition.
17   */
18  namespace org.example.basic
19
20  asset SampleAsset identified by assetId {
21    o String assetId
22    --> SampleParticipant owner
23    o String value
24  }
25
26  participant SampleParticipant identified by participantId {
27    o String participantId
28    o String firstName
29    o String lastName
30  }
31
32  transaction SampleTransaction {
33    --> SampleAsset asset
34    o String newValue
35  }
36
37  event SampleEvent {
38    --> SampleAsset asset
39    o String oldValue
40    o String newValue
41  }
42
```

Script File



```
 */

/* global getAssetRegistry getFactory emit */

/**
 * Sample transaction processor function.
 * @param {org.example.basic.SampleTransaction} tx The sample transaction instance.
 * @transaction
 */
async function sampleTransaction(tx) {  // eslint-disable-line no-unused-vars

    // Save the old value of the asset.
    const oldValue = tx.asset.value;

    // Update the asset with the new value.
    tx.asset.value = tx.newValue;

    // Get the asset registry for the asset.
    const assetRegistry = await getAssetRegistry('org.example.basic.SampleAsset');
    // Update the asset in the asset registry.
    await assetRegistry.update(tx.asset);

    // Emit an event for the modified asset.
    let event = getFactory().newEvent('org.example.basic', 'SampleEvent');
    event.asset = tx.asset;
    event.oldValue = oldValue;
    event.newValue = tx.newValue;
    emit(event);
}
```

```
 * Sample access control list.
 */
rule EverybodyCanReadEverything {
    description: "Allow all participants read access to all resources"
    participant: "org.example.basic.SampleParticipant"
    operation: READ
    resource: "org.example.basic.*"
    action: ALLOW
}

rule EverybodyCanSubmitTransactions {
    description: "Allow all participants to submit transactions"
    participant: "org.example.basic.SampleParticipant"
    operation: CREATE
    resource: "org.example.basic.SampleTransaction"
    action: ALLOW
}

rule OwnerHasFullAccessToTheirAssets {
    description: "Allow all participants full access to their assets"
    participant(p): "org.example.basic.SampleParticipant"
    operation: ALL
    resource(r): "org.example.basic.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
}

rule NetworkAdminUser {
    description: "Grant business network administrators full access to user resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "**"
    action: ALLOW
}

rule NetworkAdminSystem {
    description: "Grant business network administrators full access to system resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}
```

FILES

About
README.md, package.json

Model File
models/sample.cto

Script File
lib/sample.js

Access Control
permissions.acl

Add a file... Export

UPDATE NETWORK

From: *0.2.6-20180530153450*

To: *0.2.6-deploy.0*

Deploy changes

About File README.md

Basic Sample Business Network

> *This is the "Hello World" of Hyperledger Composer samples, which demonstrates the core functionality of Hyperledger Composer by changing the value of an asset.*

This business network defines:

Participant `SampleParticipant`

Asset `SampleAsset`

Transaction `SampleTransaction`

Event `SampleEvent`

SampleAssets are owned by a SampleParticipant, and the value property on a SampleAsset can be modified by submitting a SampleTransaction. The SampleTransaction emits a SampleEvent that notifies applications of the old and new values for each modified SampleAsset.

To test this Business Network Definition in the **Test** tab:

Create a `SampleParticipant` participant:

```
{
  "$class": "org.example.basic.SampleParticipant",
  "participantId": "Toby",
  "firstName": "Tobias",
  "lastName": "Hunter"
}
```

Create a `SampleAsset` asset:

```
{
  "$class": "org.example.basic.SampleAsset",
  "assetId": "assetId:1",
  "owner": "resource:org.example.basic.SampleParticipant#Toby",
  "value": "original value"
}
```

Submit a `SampleTransaction` transaction:

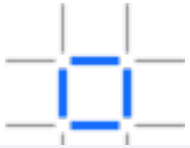
```
{
  "$class": "org.example.basic.SampleTransaction",
  "asset": "resource:org.example.basic.SampleAsset#assetId:1",
  "newValue": "new value"
}
```

After submitting this transaction, you should now see the transaction in the Transaction Registry and that a `SampleEvent` has been emitted. As a result, the value of the `assetId:1` should now be `new value` in the Asset Registry.

Participant registry for org.example.basic.SampleParticipant

+ Create New Participant

ID	Data
Toby	{ "$class": "org.example.basic.SampleParticipant", "participantId": "Toby", "firstName": "Tobias", "lastName": "Hunter" }

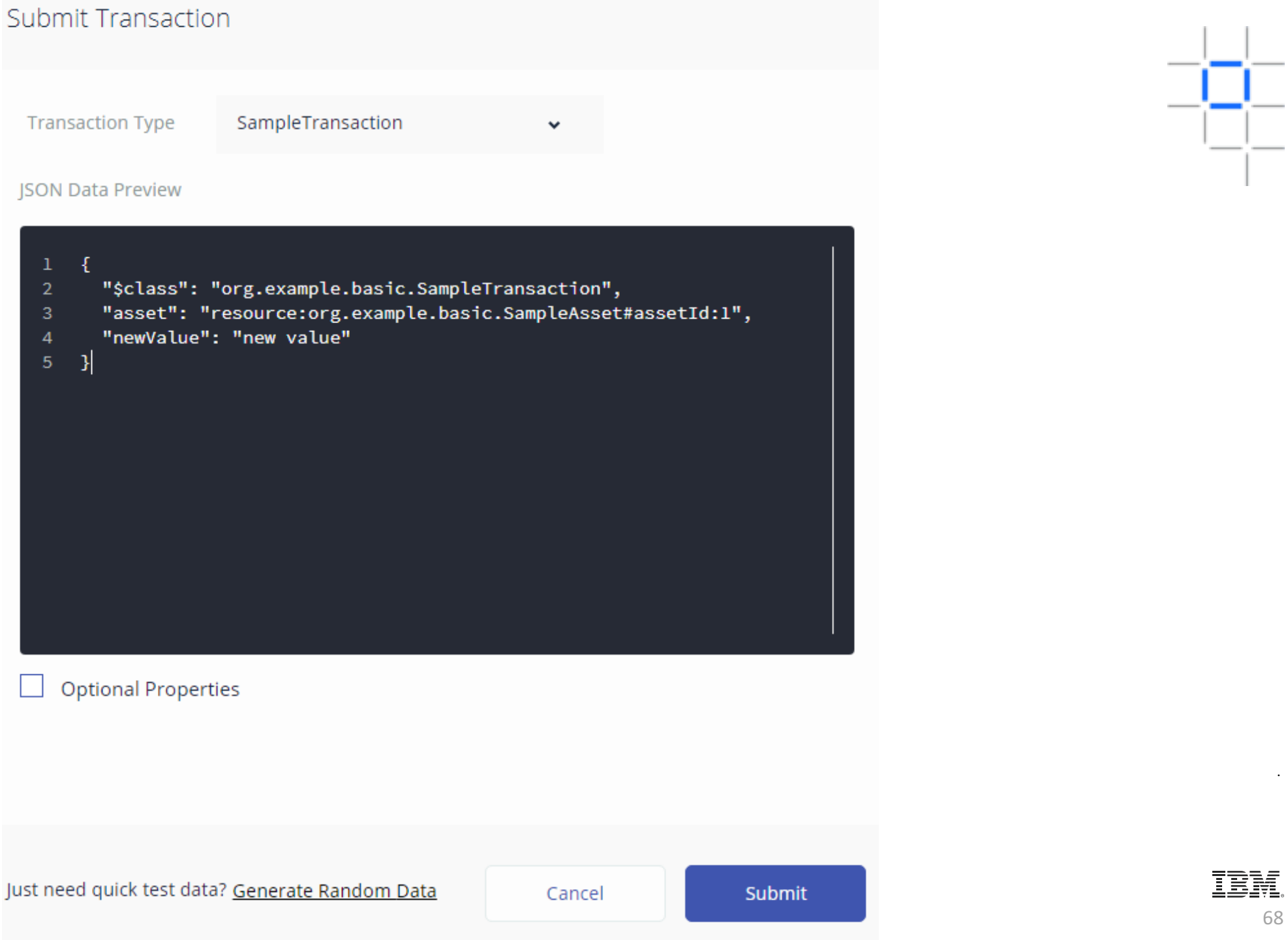


Asset registry for org.example.basic.SampleAsset

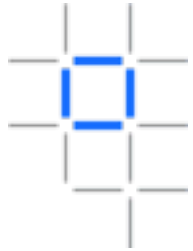
+ Create New Asset

ID	Data

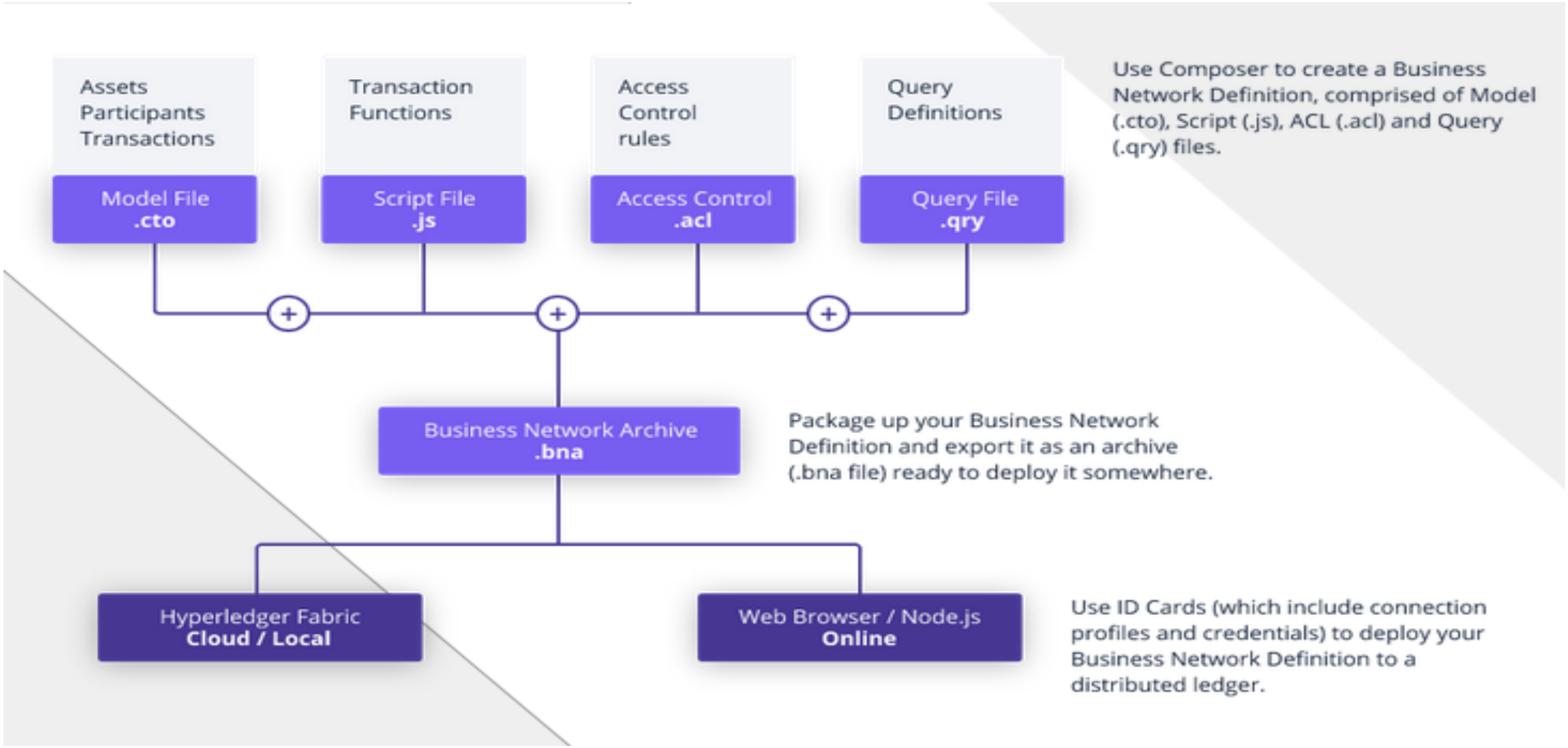
assetId:1	{
	"$class": "org.example.basic.SampleAsset",
	"assetId": "assetId:1",
	"owner": "resource:org.example.basic.SampleParticipant#Toby",
	"value": "original value"
	}


Submit Transaction
Transaction Type
SampleTransaction
JSON Data Preview
1 {
2 "$class": "org.example.basic.SampleTransaction",
3 "asset": "resource:org.example.basic.SampleAsset#assetId:1",
4 "newValue": "new value"
5 }
Optional Properties
Just need quick test data? Generate Random Data
Cancel
Submit

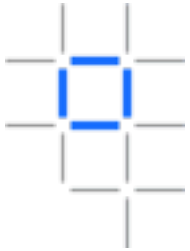

Historian Record
Transaction
Events (1)
{
"$class": "org.example.basic.SampleTransaction",
"asset": "resource:org.example.basic.SampleAsset#assetId:1",
"newValue": "new value",
"transactionId": "a100b354-4ae3-47c5-a341-d255a724fdee",
"timestamp": "2018-06-23T19:26:21.501Z"
}

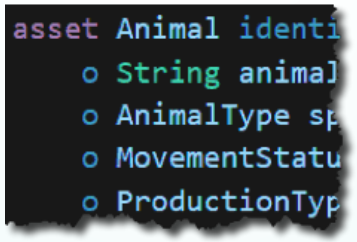


Deploying the BNA file to Hyperledger Fabric and the Cloud



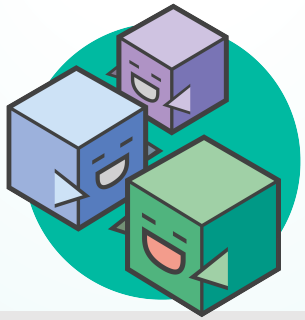
Extensive, Familiar, Open Development Toolset





Data modelling

JavaScript business logic



Web playground

Client libraries

Editor support

CLI utilities

Code generation

Existing systems and data

Let’s build an end-to-end Hyperledger application!

© 2017 IBM Corporation

Pre-reqs for Ubuntu or the Mac

The following are prerequisites for installing the required development tools:

- At least 4Gb of memory
- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (note version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- A code editor of your choice, we recommend VSCode.

https://github.ibm.com/Skills-Network/Installing-prereqs.html

Note: Nvm use node

Now using node v8.11.2 (npm v5.6.0)

© 2017 IBM Corporation

Section 1: install the Composer CLI tools and the Fabric

© 2017 IBM Corporation

1) Install Hyperledger Composer CLI tools

1. Essential CLI tools:

```
npm install -g composer-cli
```
Copy

2. Utility for running a REST Server on your machine to expose your business networks as RESTful APIs:

```
npm install -g composer-rest-server
```
Copy

3. Useful utility for generating application assets:

```
npm install -g generator-hyperledger-composer
```
Copy

4. Yeoman is a tool for generating applications, which utilises `generator-hyperledger-composer` :

Plus Yeoman

```
npm install -g yo
```
Copy

https://hyperledger.github.io/composer/latest/installing/development-tools.html

© 2017 IBM Corporation

2) Install Hyperledger Fabric

```
mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers

curl -O
https://raw.githubusercontent.com/hyperledger/composer-
tools/master/packages/fabric-dev-servers/fabric-dev-
servers.tar.gz
tar -xvf fabric-dev-servers.tar.gz
```

Copy

A `zip` is also available if you prefer: just replace the `.tar.gz` file with `fabric-dev-servers.zip` and the `tar -xvf` command with a `unzip` command in the preceding snippet.

Use the scripts you just downloaded and extracted to download a local Hyperledger Fabric runtime:

```
cd ~/fabric-dev-servers
./downloadFabric.sh
```

Copy

https://hyperledger.github.io/composer/latest/installing/development-tools.html

© 2017 IBM Corporation

Startup a new runtime

The first time you start up a new runtime, you'll need to run the start script, then generate a PeerAdmin card:

```
cd ~/fabric-dev-servers
./startFabric.sh
./createPeerAdminCard.sh
```

Copy

You can start and stop your runtime using `~/fabric-dev-servers/stopFabric.sh`, and start it again with `~/fabric-dev-servers/startFabric.sh`.

https://hyperledger.github.io/composer/latest/installing/development-tools.html

© 2017 IBM Corporation

Section 2 Let’s write our application

© 2017 IBM Corporation

Create a skeleton Business Network Archive with Yeoman

Create a skeleton business network using Yeoman. This command will require a business network name, description, author name, author email address, license selection and namespace.

yo hyperledger-composer:businessnetwork

1.Enter tutorial-network for the network name, and desired information for description, author name, and author email.
2.Select Apache-2.0 as the license.
3.Select org.example.mynetwork as the namespace.
4.Select No when asked whether to generate an empty network or not.

- https://hyperledger.github.io/composer/latest/tutorials/developer-tutorial.html

© 2017 IBM Corporation

Business Network Archive files

- CTO Model file
- JavaScript
- ACL

© 2017 IBM Corporation

Open the org.example.mynetwork.cto model file. Replace the contents with the following:

Copy

```
/**
 * My commodity trading network
 */
namespace org.example.mynetwork
asset Commodity identified by tradingSymbol {
    o String tradingSymbol
    o String description
    o String mainExchange
    o Double quantity
    --> Trader owner
}
participant Trader identified by tradeId {
    o String tradeId
    o String firstName
    o String lastName
}
transaction Trade {
    --> Commodity commodity
    --> Trader newOwner
}
```

https://hyperledger.github.io/composer/latest/tutorials/developer-tutorial.html

© 2017 IBM Corporation

Replace the javascript file

```
/**
 * Track the trade of a commodity from one trader to another
 * @param {org.example.mynetwork.Trade} trade - the trade to be processed
 * @transaction
 */
async function tradeCommodity(trade) {
    trade.commodity.owner = trade.newOwner;
    let assetRegistry = await
getAssetRegistry('org.example.mynetwork.Commodity');
    await assetRegistry.update(trade.commodity);
}
```

© 2017 IBM Corporation

Replace the acl file

```
/**
 * Access control rules for tutorial-network
 */
rule Default {
    description: "Allow all participants access to all resources"
    participant: "ANY"
    operation: ALL
    resource: "org.example.mynetwork.*"
    action: ALLOW
}

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

© 2017 IBM Corporation

Generate your Business Network Archive

From the tutorial-network directory, run the following command:

composer archive create -t dir -n .

After the command has run, a business network archive file called **tutorial-network@0.0.1.bna** has been created in the tutorial-network directory.

© 2017 IBM Corporation

Deploy the business network

```
composer network install --card PeerAdmin@hlfv1 --archiveFile tutorial-network@0.0.1.bna
```

Copy

The `composer network install` command requires a PeerAdmin business network card (in this case one has been created and imported in advance), and the the file path of the `.bna` which defines the business network.

2. To start the business network, run the following command:

```
composer network start --networkName tutorial-network --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkadmin.card
```

Copy

The `composer network start` command requires a business network card, as well as the name of the admin identity for the business network, the name and version of the business network and the name of the file to be created ready to import as a business network card.

3. To import the network administrator identity as a usable business network card, run the following command:

```
composer card import --file networkadmin.card
```

Copy

The `composer card import` command requires the filename specified in `composer network start` to create a card.

4. To check that the business network has been deployed successfully, run the following command to ping the network:

```
composer network ping --card admin@tutorial-network
```

Copy

Business Network Cards

- A Business Network Card provides all of the information needed to connect to a blockchain business network.
- It is only possible to access a blockchain Business Network through a valid Business Network Card.
- A Business Network Card contains an Identity for a single Participant within a deployed business network.
- Business Network Cards are used in the Hyperledger Composer Playground to connect to deployed Business Networks.

https://hyperledger.github.io/composer/v0.16/playground/id-cards-playground

© 2017 IBM Corporation

Section 3 Install Rest Server and Angular frontend and run our application – thank you Ruby on Rails!

© 2017 IBM Corporation

Install the Composer Rest Server on port 3000

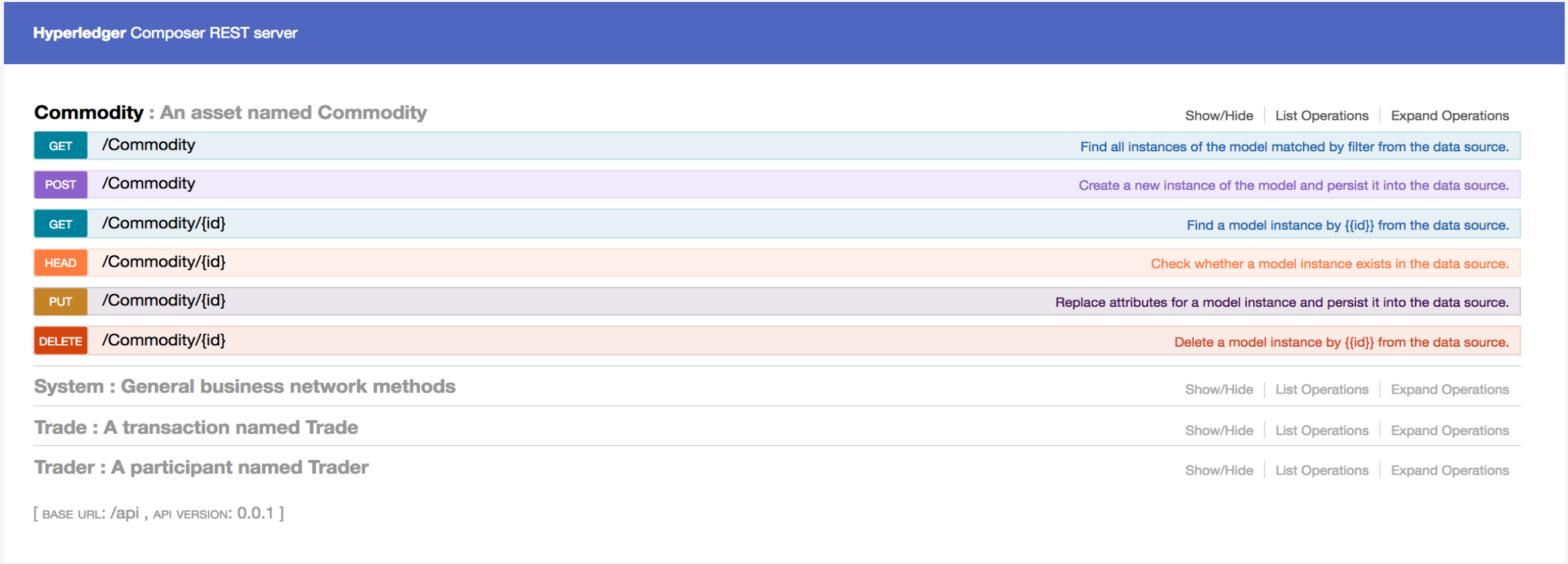
1. To create the REST API, navigate to the `tutorial-network` directory and run the following command:

```
composer-rest-server
```

Copy

© 2017 IBM Corporation

Port 3000



© 2017 IBM Corporation

Generate the Angular Application

Step Six: Generating an application

Hyperledger Composer can also generate an Angular 4 application running against the REST API.

1. To create your Angular 4 application, navigate to `tutorial-network` directory and run the following command:

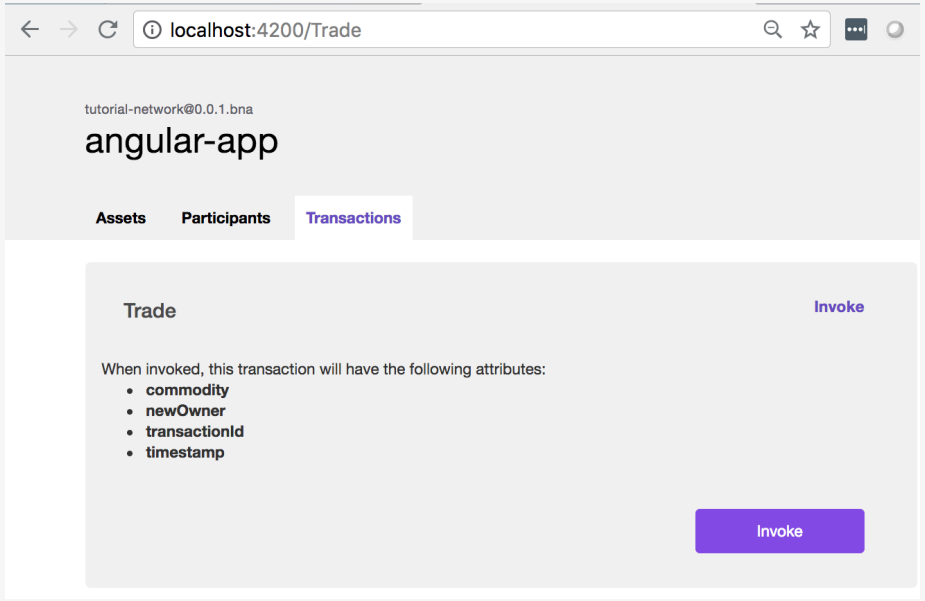
```
yo hyperledger-composer:angular
```

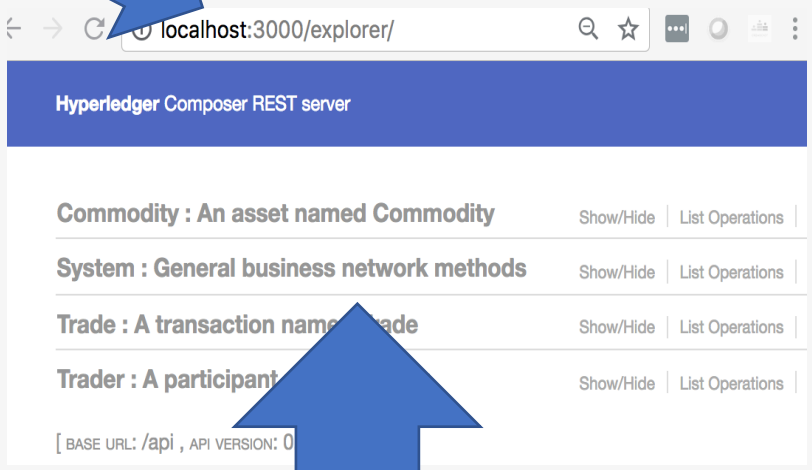
Copy

The Angular generator will then create the scaffolding for the project and install all dependencies. To run the application, navigate to your angular project directory and run **npm start .** This will fire up an Angular 4 application running against your REST API at http://localhost:4200 .

© 2017 IBM Corporation

The Hyperledger App, REST server and Angular app





Hyperledger App

© 2017 IBM Corporation

Angular app 1

tutorial-network@0.0.1.bna

angular-app

Assets **Participants** **Transactions**

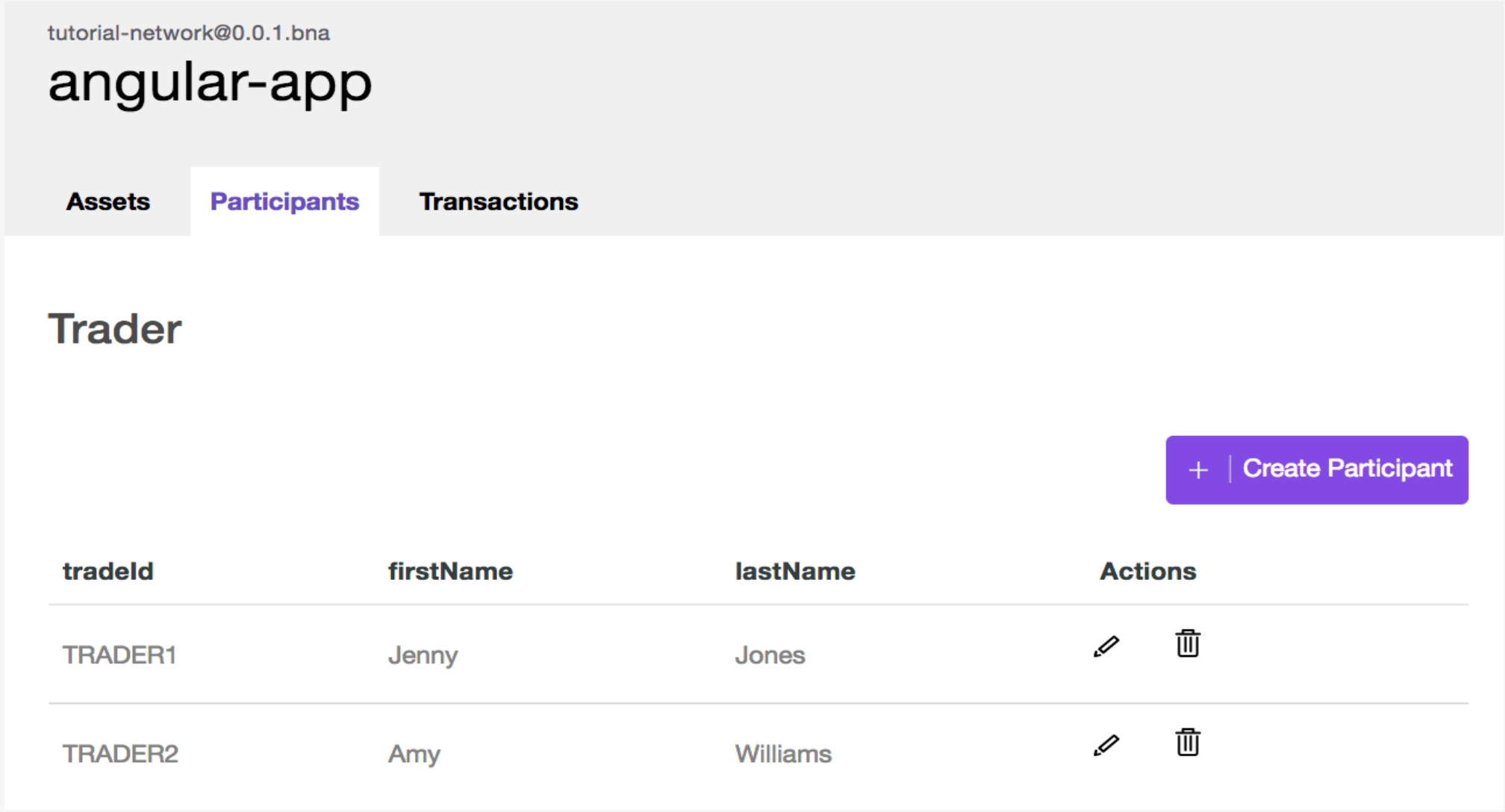
Commodity

+ | Create Asset

tradingSymbol	description	mainExchange	quantity	owner	Actions
ABC	Test Commodity	Euronext	72.297	resource:org.example.mynetwork.T…	

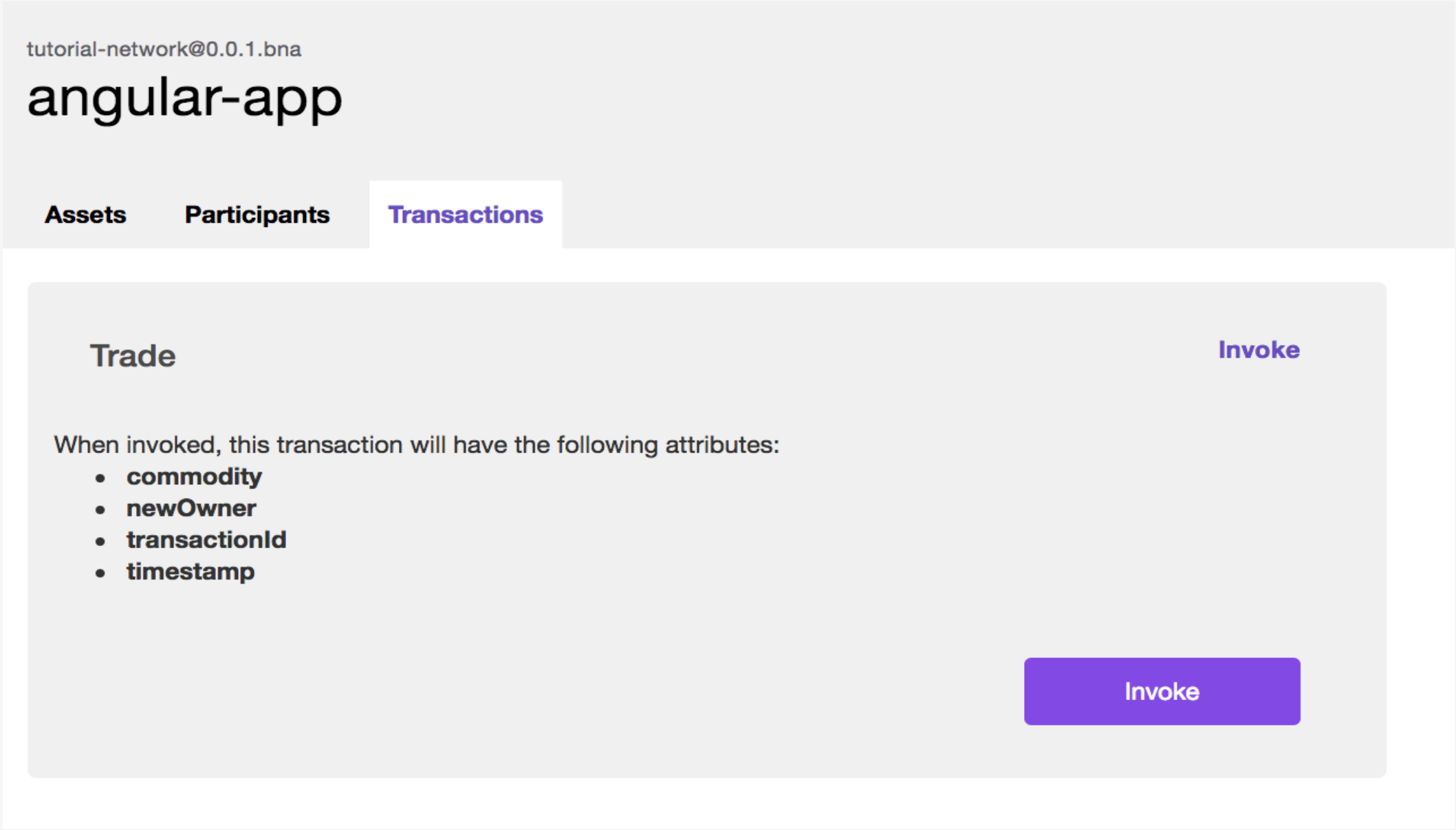
© 2017 IBM Corporation

Angular App 2

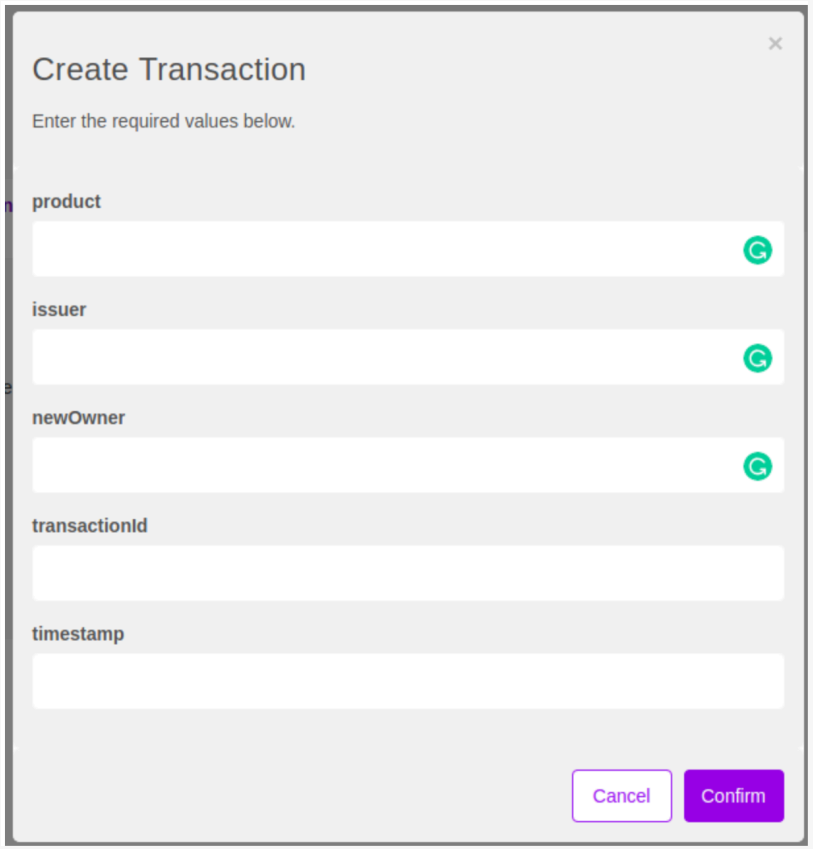


© 2017 IBM Corporation

Angular App 3



Fixing Transaction issue in Angular for Hyperledger fabric blockchain application



https://medium.com/coinmonks/fixing-transaction-issue-in-angular-for-hyperledger-fabric-blockchain-application-fe7e28a7bb6e

© 2017 IBM Corporation

Fixing Transaction issue in Angular for Hyperledger fabric blockchain application

Yashwanth Madaka

https://medium.com/coinmonks/fixing-transaction-issue-in-angular-for-hyperledger-fabric-blockchain-application-fe7e28a7bb6e

Building a blockchain application using Hyperledger Fabric with Angular Frontend: Part-2

https://medium.com/coinmonks/building-a-blockchain-application-using-hyperledger-fabric-with-angular-frontend-part-2-22ef7c77f53

https://medium.freecodecamp.org/ultimate-end-to-end-tutorial-to-create-an-application-on-blockchain-using-hyperledger-3a83a80cbc71

© 2017 IBM Corporation

The business network and complete applications

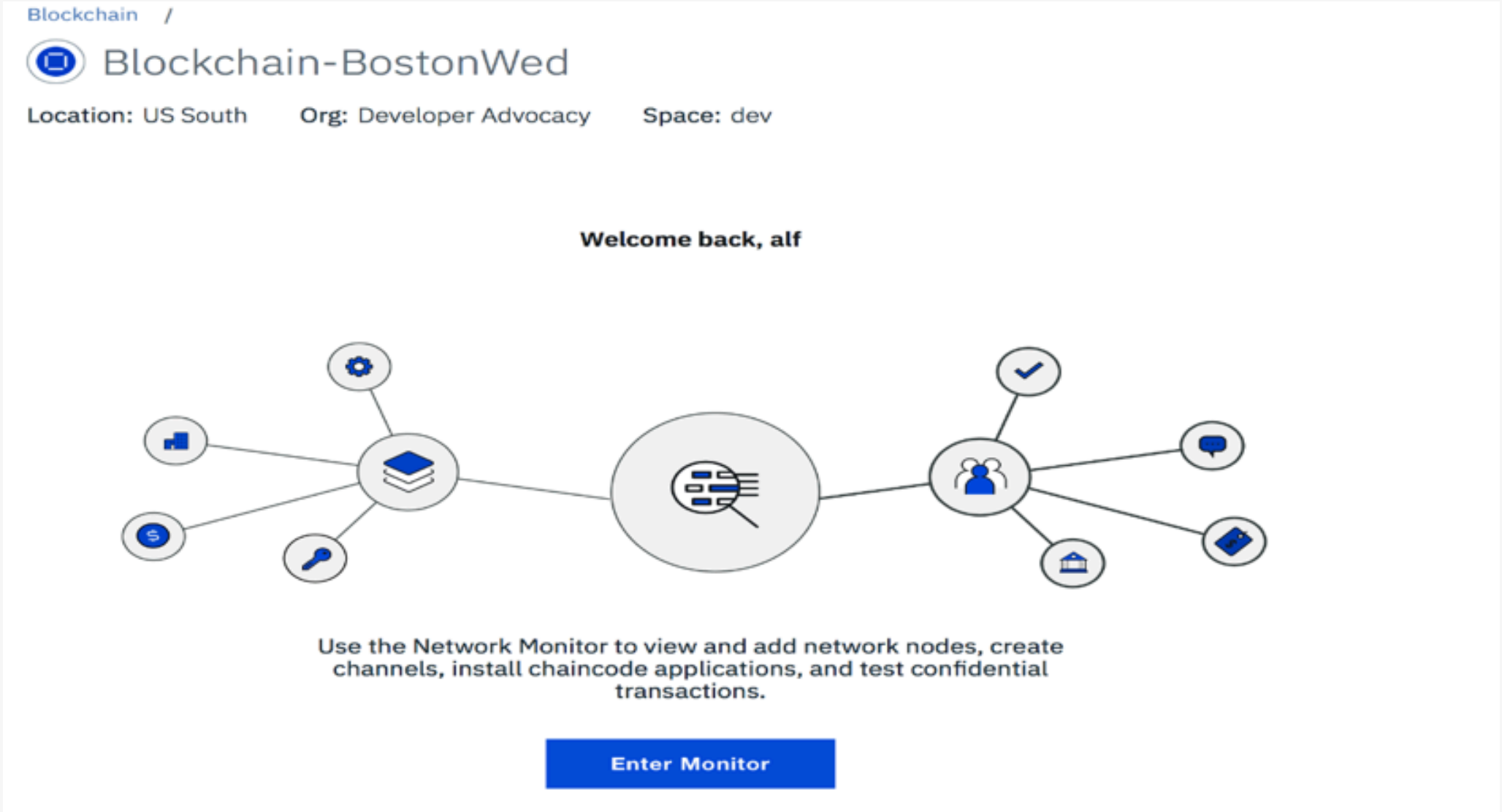
https://developer.ibm.com/code/patterns/decentralized-energy-hyperledger-composer/

© 2017 IBM Corporation

Step 4: Deploy Hyperledeger apps in the cloud

© 2017 IBM Corporation

Deploy Hyperledger apps in the cloud



© 2017 IBM Corporation

Run Blockchain apps in the IBM Cloud

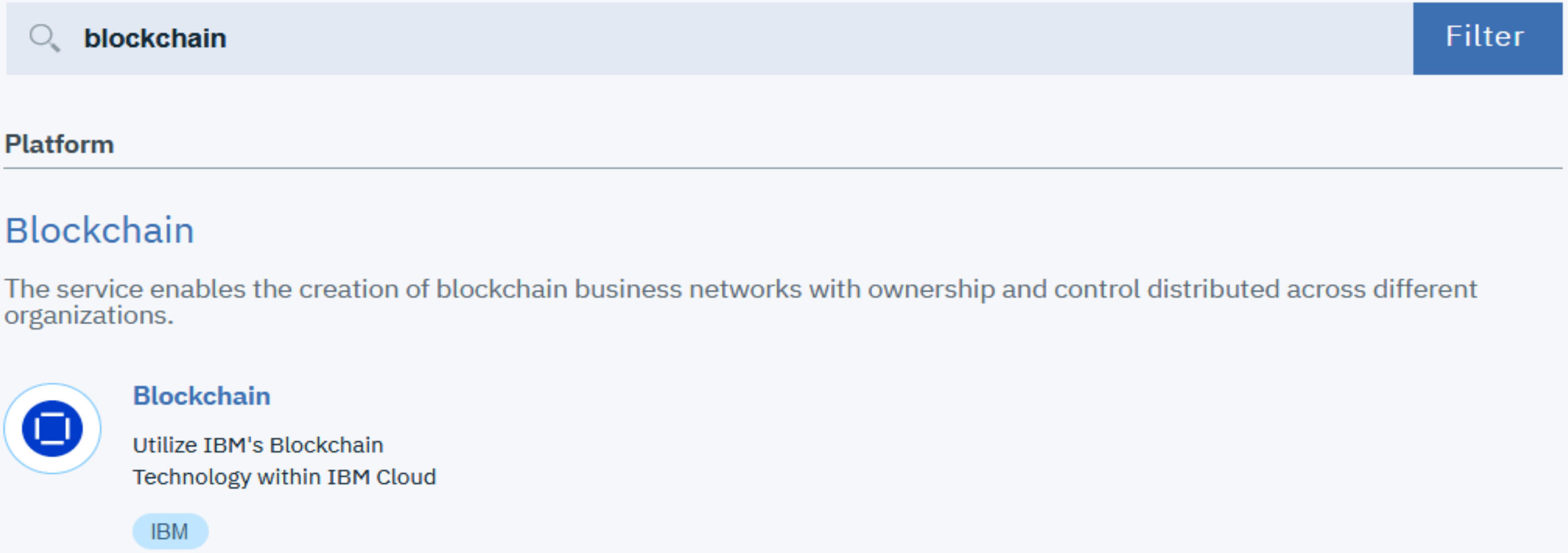
- https://vehicle-manufacture-20180324185808931.mybluemix.net/tutorial

- **vehicle-manufacture-uneradicative-insolence.mybluemix.net**
- **http://vehicle-manufacture-uneradicative-insolence.mybluemix.net/tutorial**

© 2017 IBM Corporation

How do we host our Blockchain application?

IBM Cloud Catalog



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Blockchain

IBM Blockchain Platform is a flexible software-as-a-service offering that is delivered via the IBM Cloud. It enables network members to quickly get started developing and easily move to a collaborative environment. The platform simplifies your blockchain journey of developing, governing, and operating a network. Choose a membership plan based on your ecosystem needs.

IBM

View Docs Terms

AUTHOR IBM

PUBLISHED 07/05/2018

Features

- **Use the IBM Blockchain Platform to simplify the developmental, governmental, and operational aspects of creating a blockchain solution. The following plans enable you to easily migrate from POC to pilot, all the way through to production on a secure, high performance, and fully scalable production network that you can't outgrow.**
- **For more information on developing, governing, and operating your blockchain network, see https://ibm.biz /bcdocumentation.**
- **Get $500 towards your first network with Starter Plan, featuring an easy-to-use UI to reduce network administration and governance time, an iterative development platform and basic service levels for pilot evaluation or pre-production POCs.***
- **Enterprise Plan offers a secure production environment and advanced service levels for production grade deployment, application development, and production testing.**

Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the

Need Help?
Contact IBM Cloud Sales

Already have an account?
Log in

Sign up to Create

Matterhorn Network - IPU

MY NETWORK

- Overview
- Members
- Channels
- Notifications
- APIs

MY CODE

- Develop code
- Install code
- Try samples

Overview

View and manage network resources for your organization. You can stop or start your resources and view the log file of the resource by selecting View Logs under "Actions". Learn more

Connection Profile | Add Peers

TYPE	NAME	STATUS	ACTIONS
Orderer	orderer	Running	
CA	org1-ca	Running	
Peer	org1-peer1	Running	

Members

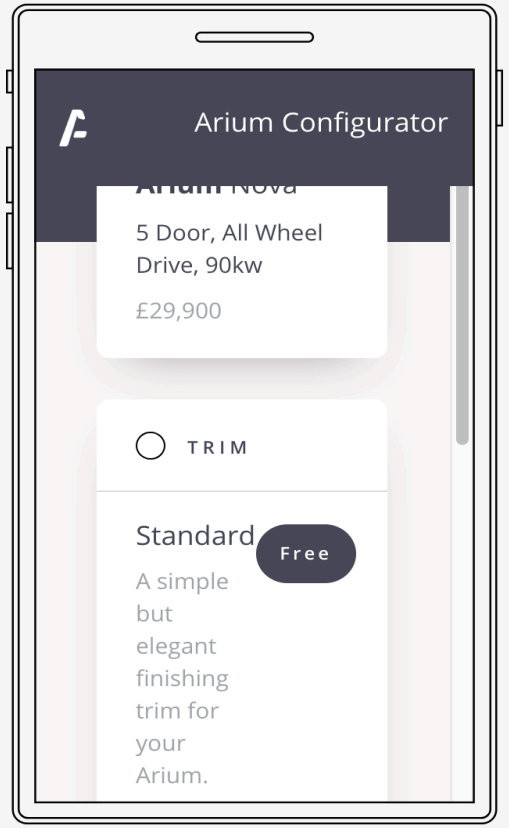
MEMBERS (2/15)	MSP ID	REQUESTER	STATUS
Company A alf@us.ibm.com	org1	--	Joined
Company B alf@us.ibm.com	org2	--	Joined

Channels

Search Channels

ID	TIME CREATED	BLOCK HEIGHT	PEERS	A
defaultchannel	07/18/18 06:08 PDT	1	org1-peer1	

Run Blockchain apps in the IBM Cloud



HYPERLEDGER
BLOCKCHAIN TECHNOLOGIES FOR BUSINESS

Thank You!

alf@us.ibm.com

https://hyperledger.org

https://ibm.com/blockchain

Spectrum of Blockchains

Permissionless Public	Permissionless Private	Permissioned Public	Permissioned Private
Bitcoin, Ethereum	Public Polls	Land Titles, University Degrees	Medical Records Business Networks

Permissioned vs. Permissionless: Who can write to a Blockchain, i.e. accessibility
Public vs Private: who can read from a Blockchain, i.e. visibility

https://www.youtube.com/watch?v=-O3ouBdG3Xg