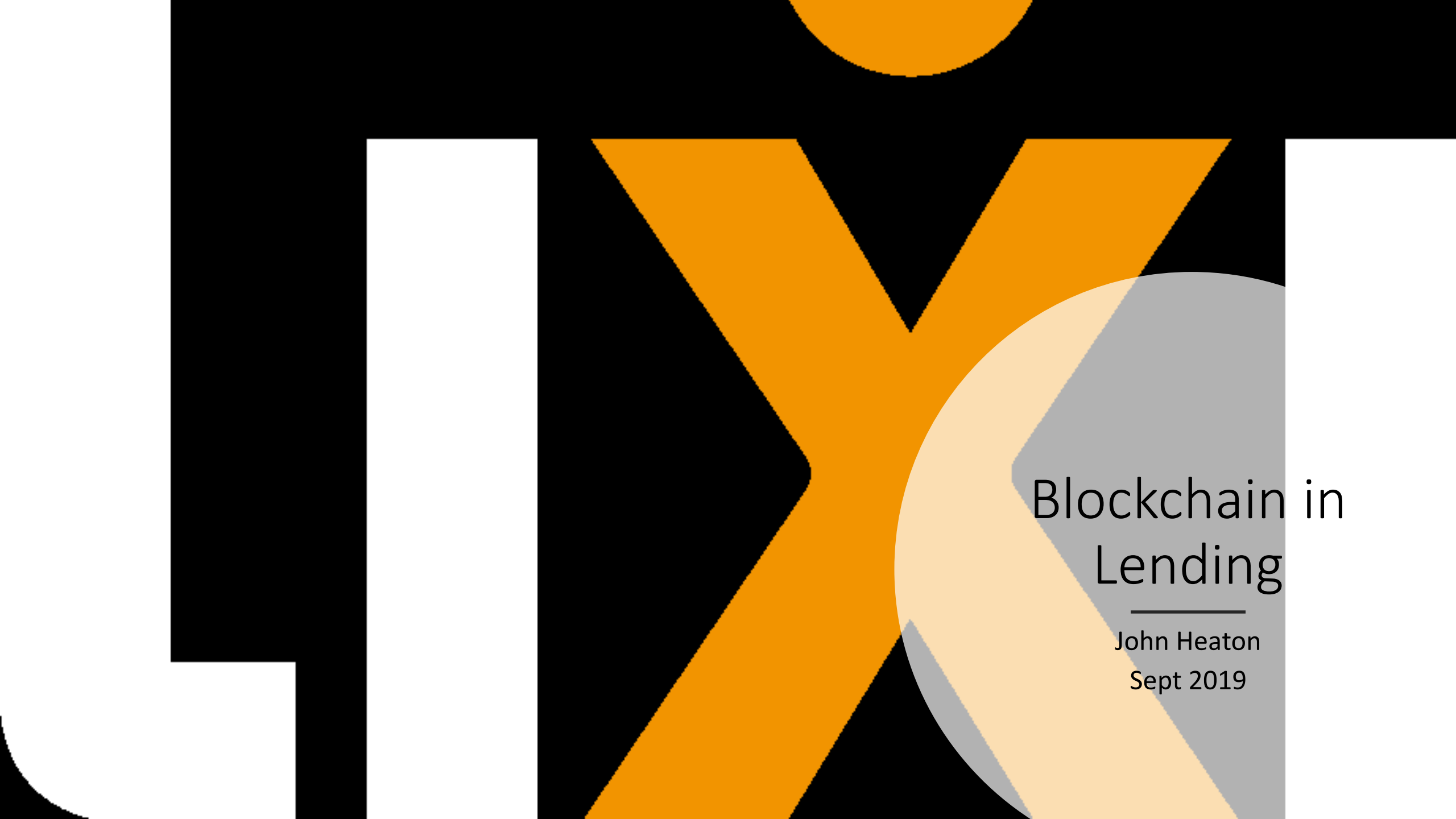




Blockchain in Lending

John Heaton
Sept 2019

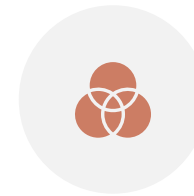
Blockchain Value Drivers



Operational simplification:
DLT reduces manual efforts
required to perform
reconciliation and resolve
disputes.



Regulatory efficiency
improvement: DLT enables
real-time monitoring by
regulators of the financial
activity of regulated entities.



Counterparty risk reduction:
DLT challenges the need to
trust counterparties to fulfil
obligations as agreements
are codified and executed in
a shared, immutable
environment.



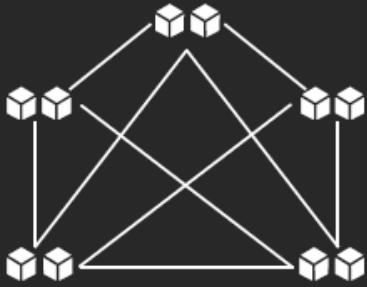
Clearing and settlement time
reduction: DLT
disintermediates third
parties that support
transaction
verification/validation and so
accelerates settlement.



Liquidity and capital
improvement: DLT reduces
locked-in capital and
provides transparency into
sourcing liquidity for assets.



Fraud minimisation: DLT
enables asset provenance
and full transaction history
to be established within a
single source of truth.



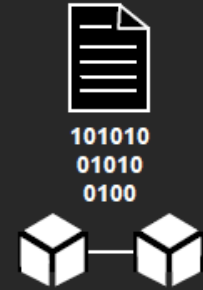
Decentralized
interactions



Provenance of data



Disintermediation



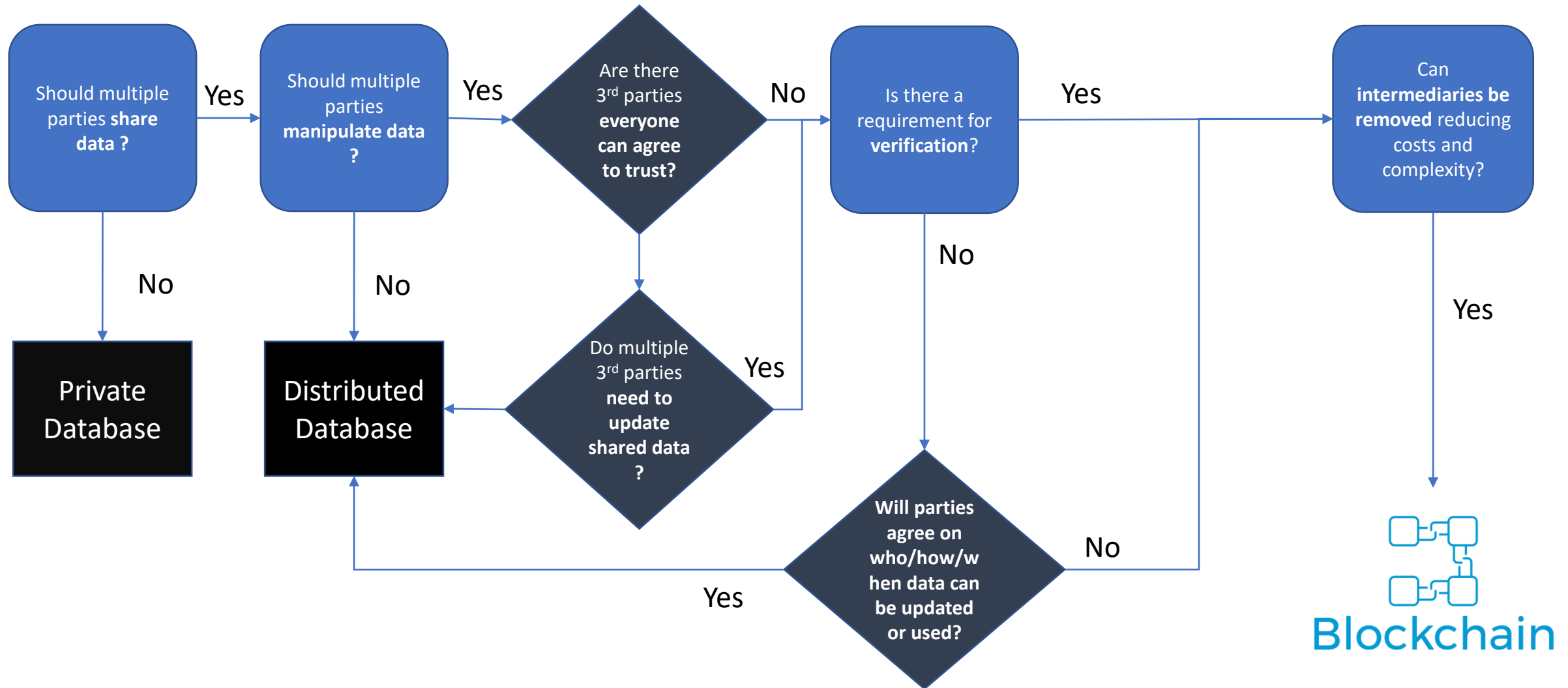
Integrated data
management

Technical Use Cases

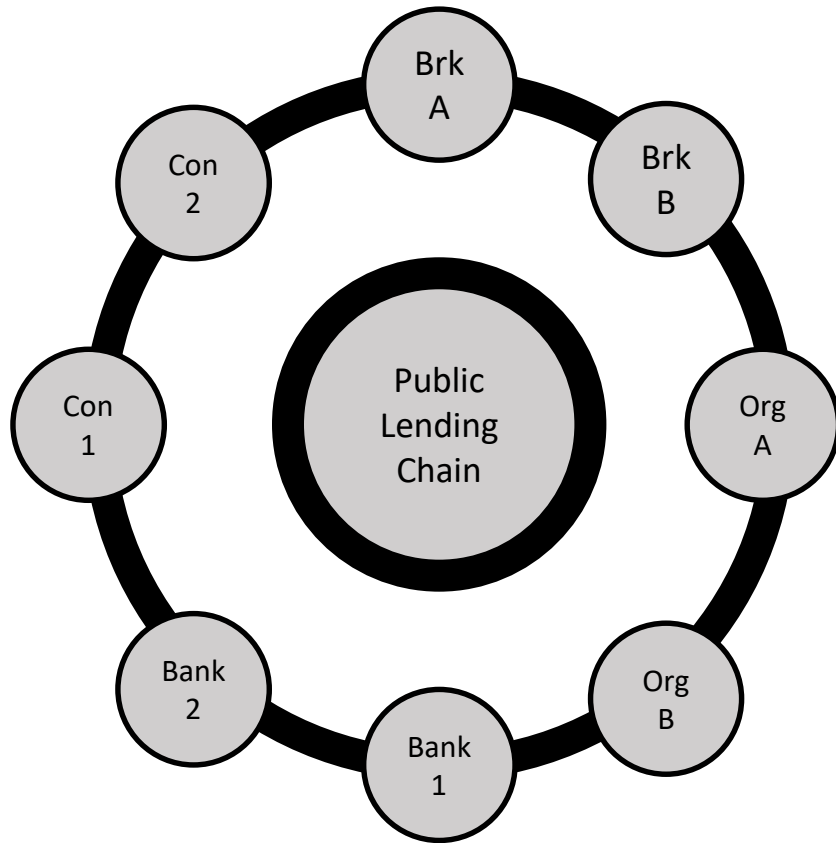
Blockchain-As-A Service

Provider	Description	Launch
Microsoft	Launched in 2016, Microsoft Blockchain on Azure is a blockchain application development platform hosted on the Microsoft Azure cloud computing platform. On this popular BaaS platform, businesses can “develop, test, and deploy secure blockchain apps.” This has helped Microsoft to attract a wide range of high-level partners such as Ripple and BitPay and has seen the platform become one of the go-to BaaS solutions for industry proofs-of-concept.	2016
IBM	Launched in 2017, the IBM Blockchain Platform is one of the powerhouses in the BaaS space. IBM Blockchain is a fully-integrated distributed ledger technology platform that enables businesses to “develop, govern, and operate a blockchain ecosystem quickly and cost effectively on a flexible, cloud-based platform.” Some of the biggest names in banking and commercial industries have already started to develop blockchain solutions using the IBM Blockchain Platform.	2017
SAP	Launched in 2017, SAP’s cloud-based blockchain service platform, SAP Leonardo, aims to empower businesses to digitally transform at scale through the use of distributed ledger technology. The platform provides plug-and-play blockchain solutions and allows for the easy setup and hosting of blockchain nodes.	2017
Amazon	In 2018, Amazon launched Blockchain on AWS, its blockchain-as-a-service solution, which enables businesses to “easily build scalable blockchain and ledger solutions.” Amazon enables businesses to build blockchain solutions on top of either Ethereum or Hyperledger Fabric as well as provide access to Amazon AWS’ other cloud computing services.	2018
Oracle	In 2017 Oracle launched its Blockchain-as-a-Service platform, called Oracle Blockchain Cloud Service, to provide an enterprise-grade distributed ledger platform that can help businesses to “increase trust and provide agility in transactions across their business networks.” Oracle Blockchain Cloud Service customers can develop blockchain solutions that can seamlessly connect with a range of popular Oracle solutions such as Oracle Supply Chain Management (SCM) Cloud, Oracle Enterprise Resource Planning (ERP) Cloud, and other Oracle cloud solutions.	2018
Hewlett-Packard	In 2017, HP launched its Blockchain-as-a-Service platform, called HPE Mission Critical Blockchain, which “enables customers to execute distributed-ledger workloads in environments that demand 100 percent fault tolerance at mission critical levels and ensures massive scalability to grow with the business as well as SQL integration with legacy systems.” The BaaS platform was developed in partnership with software company R3 and is offered on HPE Integrity NonStop platforms.	2017

Lending Business Case

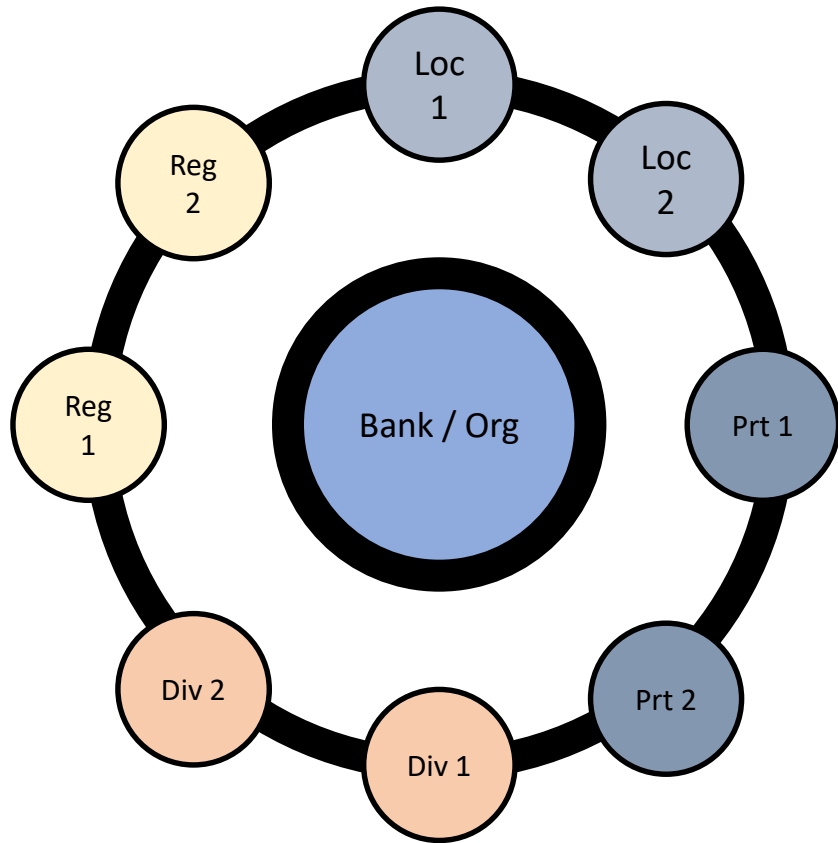


Public Lending Chain



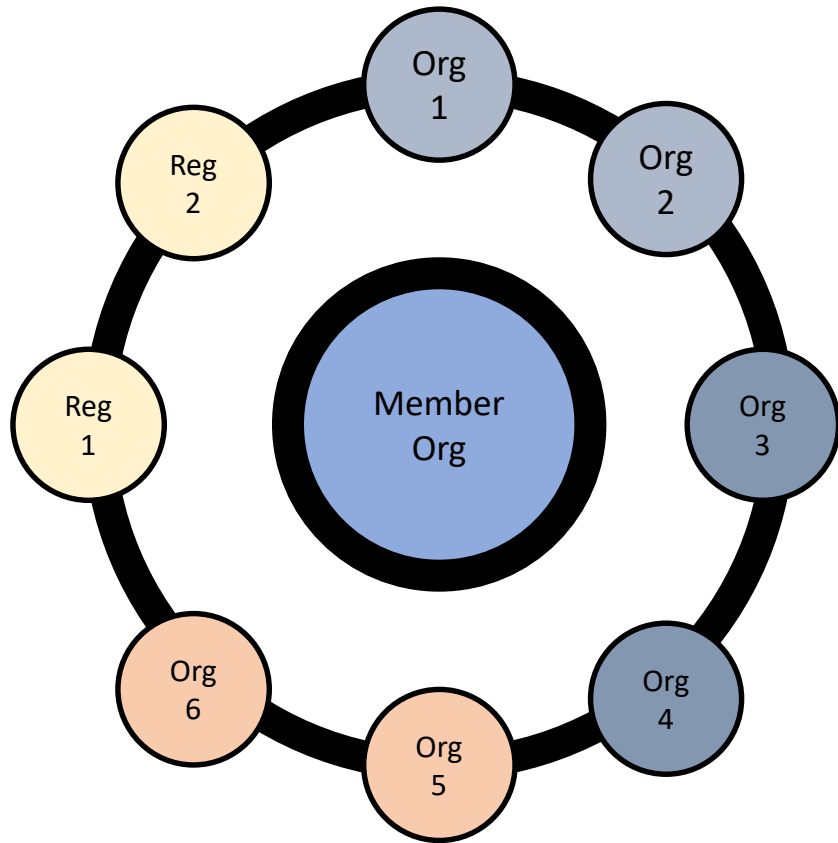
- Public Blockchain
- No centralised control authority
- Consensus algorithm validates read / write
- Proposals are submitted for voting to change the network or participants in the network

Private Lending Chain



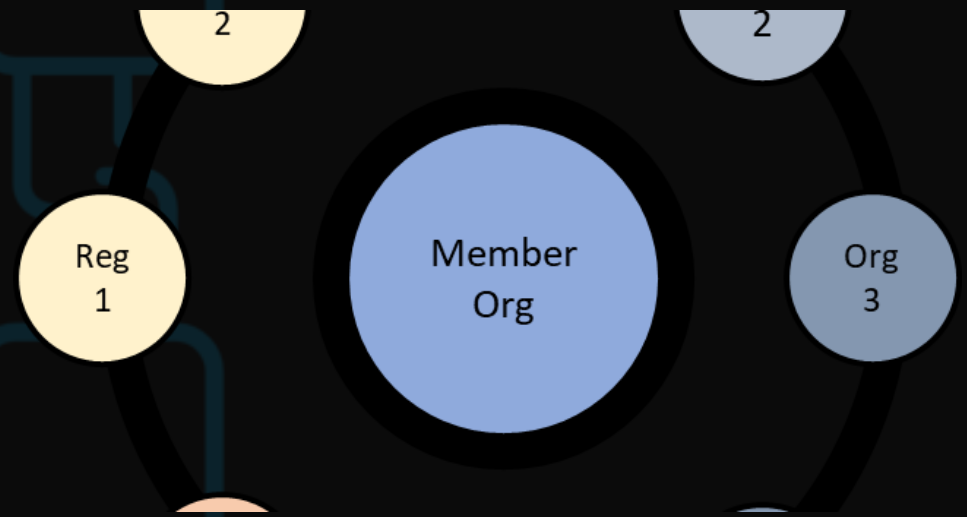
- Private Permissioned Blockchain
- Centralised control authority
- Consensus algorithm validates read / write
- Centralised authority controls changes to chain

Consortium Lending Chain



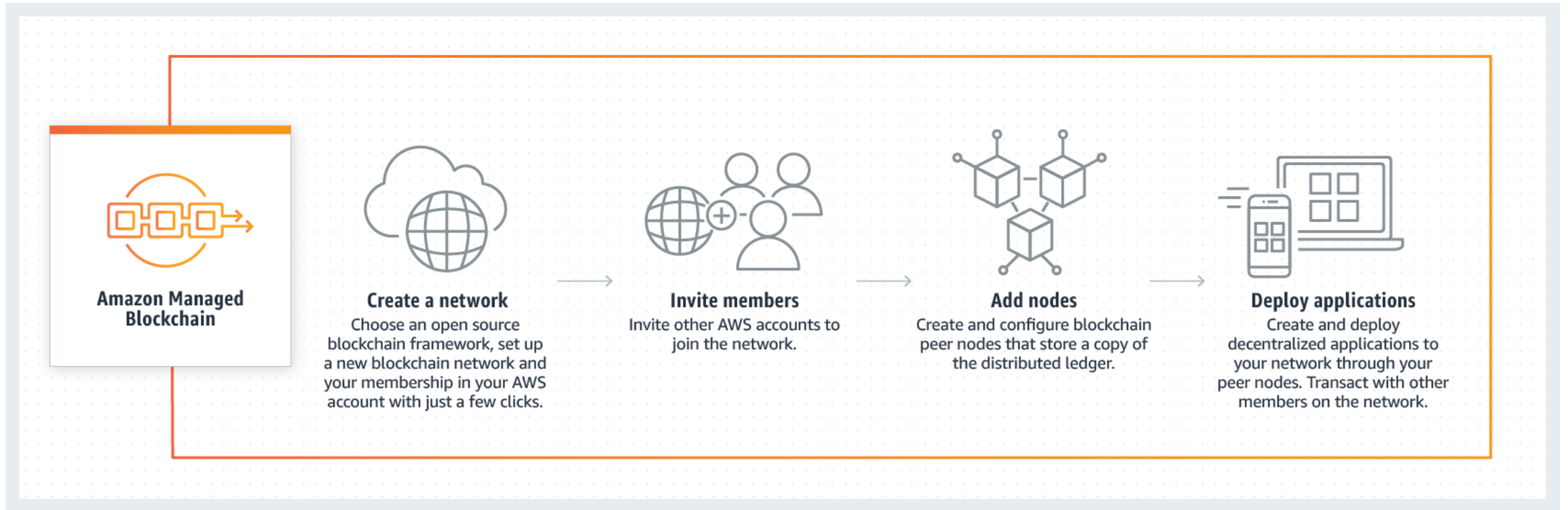
- Private Permissioned Blockchain
- Standards through centralised agency
- Members span eco-system stakeholders
- Consensus algorithm validates read / write
- Proposals are submitted for voting to change the network or participants in the network

Hypotehsis



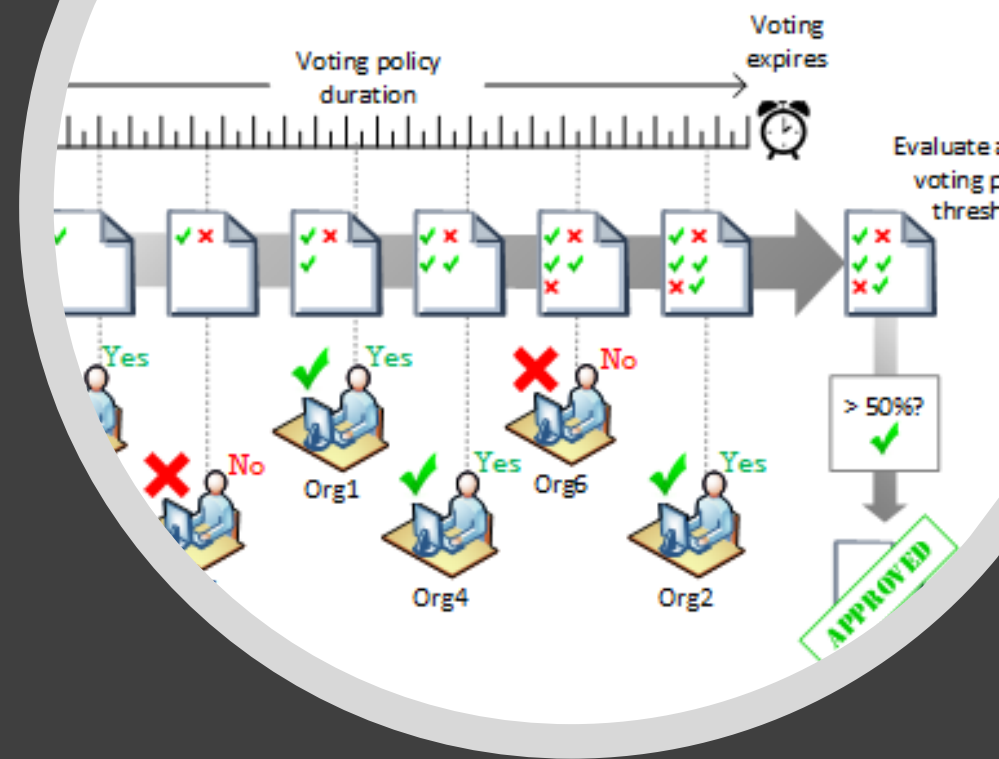
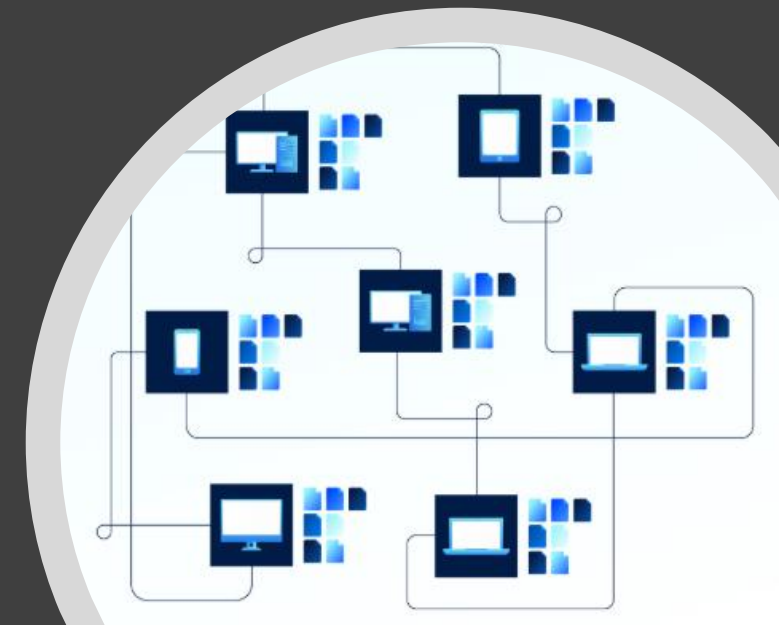
LIXI Sponsors the creation of an AWS Blockchain Network for the efficient processing of Loan Applications amongst its members

AWS Blockchain



Step 1: Create the Network and First Member

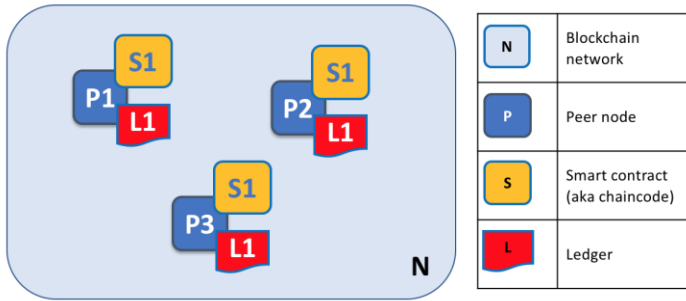
- Create the Blockchain network and specify the voting policy
- Select the blockchain framework
e.g. Hyperledger Fabric
- Create the first member



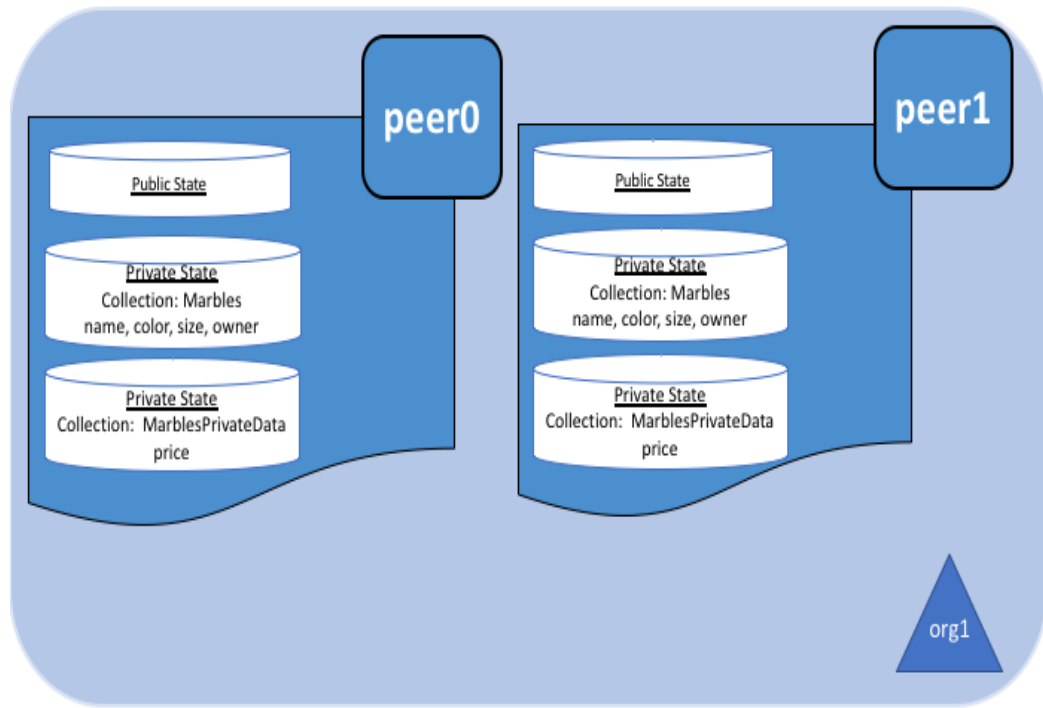
Step 2: Create and Configure the Network Interface

-
- Configure the physical network for the Blockchain network
  - Used to connect you Blockchain Client to the Blockchain Network





Step 3: Create Peer Node



Step 4: Create a Channel

A channel is a private “subnet” of communication between two or more specific network members, for the purpose of conducting private and confidential transactions.

Step 5: Install and Run Chaincode

Executable File | 865 lines (735 sloc) | 27.9 KB

Raw

Blame

History



```
1  /*
2   * SPDX-License-Identifier: Apache-2.0
3   */
4
5  'use strict';
6
7  const { Contract } = require('fabric-contract-api');
8
9  // predefined order states
10 const orderStatus = {
11   Created: {code: 1, text: 'Order Created'},
12   Bought: {code: 2, text: 'Order Purchased'},
13   Cancelled: {code: 3, text: 'Order Cancelled'},
14   Ordered: {code: 4, text: 'Order Submitted to Provider'},
15   ShipRequest: {code: 5, text: 'Shipping Requested'},
16   Delivered: {code: 6, text: 'Order Delivered'},
17   Delivering: {code: 15, text: 'Order being Delivered'},
```

What is Chaincode?

Chaincode is a program, written in [Go](#), [node.js](#), or [Java](#) that implements a prescribed interface and are executed on Docker containers.

Chaincode
Lifecycle
Status

```
Created: {code: 1, text: 'Order Created'},  
Bought: {code: 2, text: 'Order Purchased'},  
Cancelled: {code: 3, text: 'Order Cancelled'},  
Ordered: {code: 4, text: 'Order Submitted for Order'},  
ShipRequest: {code: 5, text: 'Shipping Requested'},  
Delivered: {code: 6, text: 'Order Delivered'},  
Delivering: {code: 15, text: 'Order being Delivered'},  
Backordered: {code: 7, text: 'Order Backordered'},  
Dispute: {code: 8, text: 'Order Disputed'},  
Resolve: {code: 9, text: 'Order Dispute Resolved'},  
PayRequest: {code: 10, text: 'Payment Requested'},  
Authorize: {code: 11, text: 'Payment Approved'},  
Paid: {code: 14, text: 'Payment Processed'}
```

Example : <https://developer.ibm.com/patterns/build-a-global-finance-application-on-blockchain/>


```
ance contract
```

```
inance extends Contract {
```

```
tiate with keys to collect participant ids
```

```
tantiate(ctx) {
```

```
emptyList = [];
```

```
ctx.stub.putState('buyers', Buffer.from(JSON.stringify(emptyList)));
```

```
ctx.stub.putState('sellers', Buffer.from(JSON.stringify(emptyList)));
```

```
ctx.stub.putState('shippers', Buffer.from(JSON.stringify(emptyList)));
```

```
ctx.stub.putState('providers', Buffer.from(JSON.stringify(emptyList)));
```

```
ctx.stub.putState('financeCos', Buffer.from(JSON.stringify(emptyList)));
```

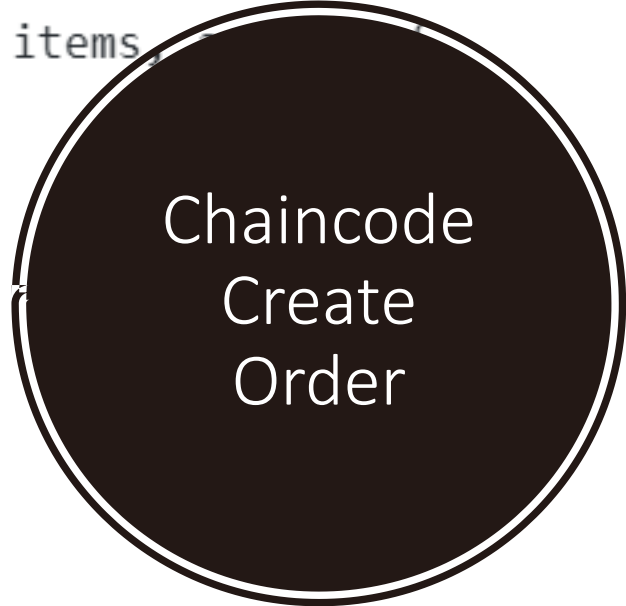


Chaincode
Define
Participants

Example : <https://developer.ibm.com/patterns/build-a-global-finance-application-on-blockchain/>

```
// add an order object to the blockchain state identified by the orderNumber
```

```
async CreateOrder(ctx, buyerId, sellerId, financeCoId, orderNumber, items,
```



Chaincode
Create
Order

```
// verify buyerId
```

```
let buyerData = await ctx.stub.getState(buyerId);
```

```
let buyer;
```

```
if (buyerData) {
```

```
    buyer = JSON.parse(buyerData.toString());
```

```
    if (buyer.type !== 'buyer') {
```

```
        throw new Error('buyer not identified');
```

```
    }
```

```
} else {
```

```
    throw new Error('buyer not found');
```

```
}
```

```
// verify sellerId
```

```
let sellerData = await ctx.stub.getState(sellerId);
```

```
let seller
```

Example : <https://developer.ibm.com/patterns/build-a-global-finance-application-on-blockchain/>

```
// add an order object to the blockchain state identified by the orderNumber
```

```
async CreateOrder(ctx, buyerId, sellerId, financeCoId, orderNumber, items,
```



Chaincode
Events &
Verify

```
// verify buyerId
```

```
let buyerData = await ctx.stub.getState(buyerId);
```

```
let buyer;
```

```
if (buyerData) {
```

```
    buyer = JSON.parse(buyerData.toString());
```

```
    if (buyer.type !== 'buyer') {
```

```
        throw new Error('buyer not identified');
```

```
    }
```

```
} else {
```

```
    throw new Error('buyer not found');
```

```
}
```

```
// verify sellerId
```

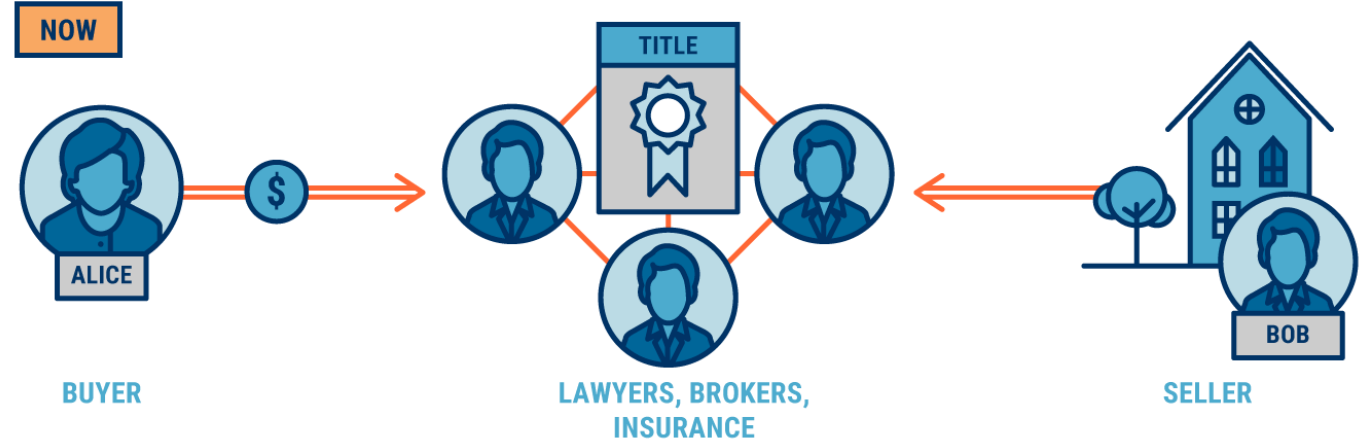
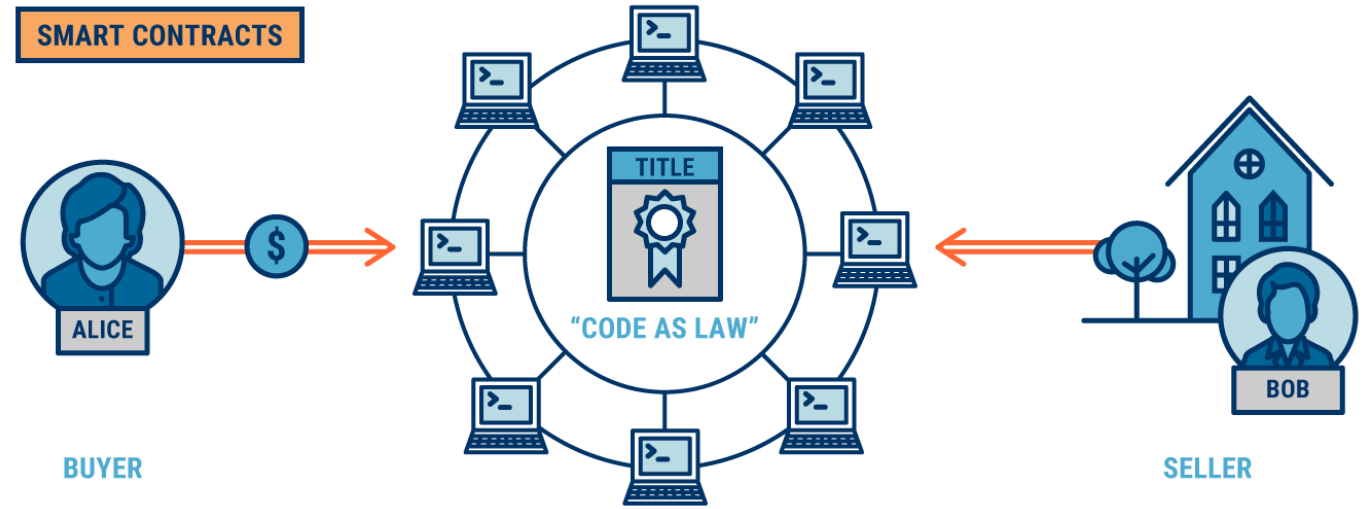
```
let sellerData = await ctx.stub.getState(sellerId);
```

```
let seller
```

Example : <https://developer.ibm.com/patterns/build-a-global-finance-application-on-blockchain/>

Chaincode & Lending

Smart Contracts



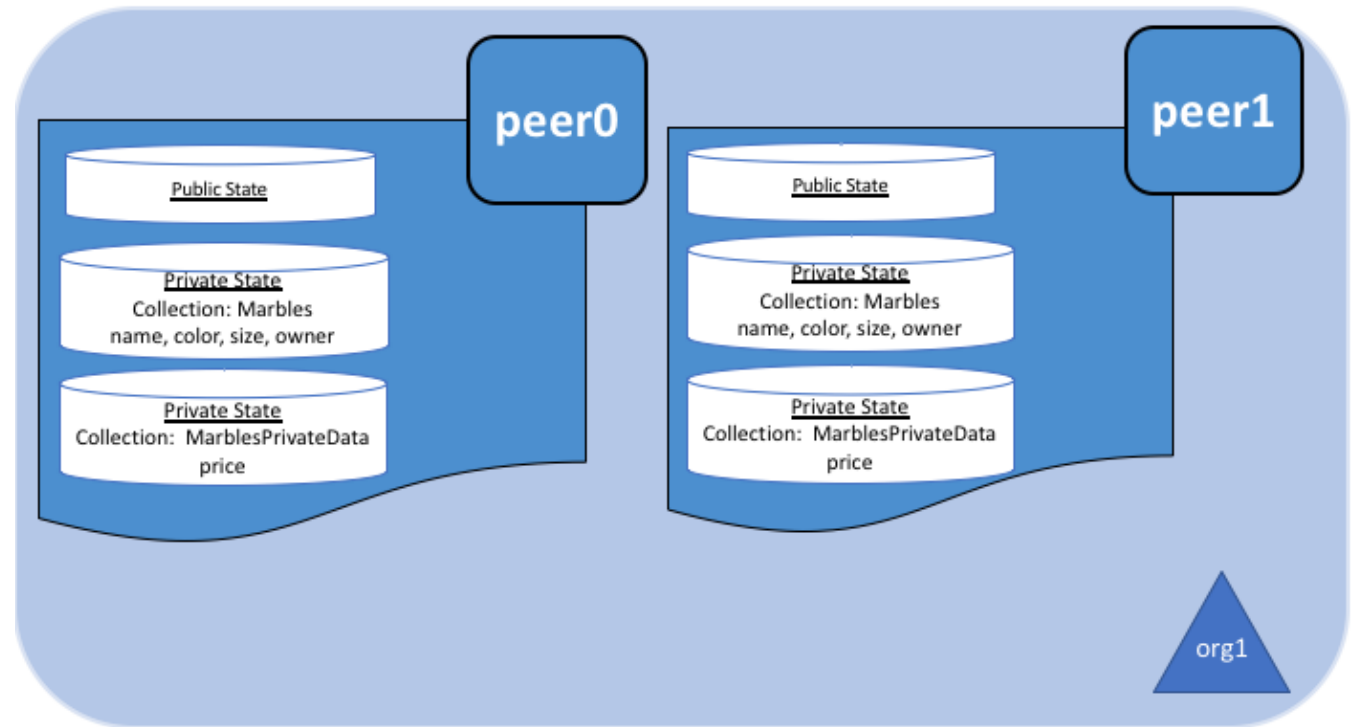
Smart Contract
Applications

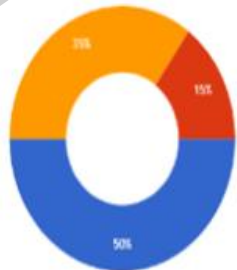
Smart Contract
Security

LIXI & Smart Contracts

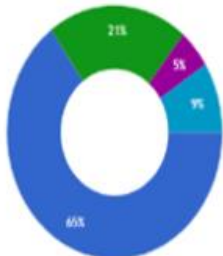
Step 6: Invite a Member and Create a Joint Channel

- Private Data Collections: A way to keep certain data/transactions confidential among a subset of channel members.
- Access control: How to configure which client identities can interact with peer functions on a per channel basis.





<6x 4-6x >6x



Term Loans - Owner Occupied Term Loans - Investment
Term Loans - Non-housing purposes Revolving Credit



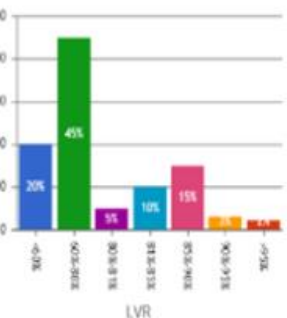
Direct Broker

12

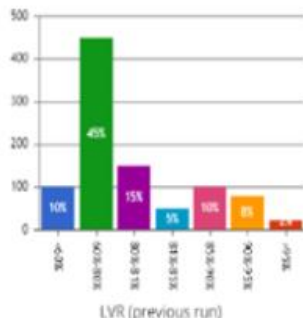
13

14

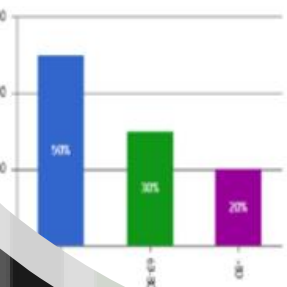
VARIANCES



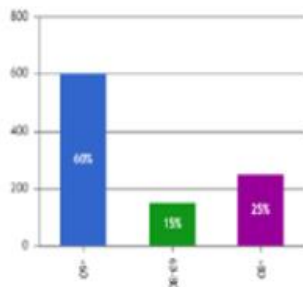
LVR



LVR (previous run)



Application Credit Worthiness



Application Credit Worthiness (previous run)

STATUS TRACKER

Category	Limit	Actual
LVR >60%	20% Min	10%
LVR 60%-80%	20% Min	25%
LVR 80%-90%	10% Max	8%
LVR 90%-95%	10% Max	
LVR 95%-99%	10% Max	
LVR >99%	10% Max	
Application Credit Worthiness		
Application Credit Worthiness		

Step 7: Deploy Applications

Loan Product

of Hyperledger Fabric Hosts

6

of Docker Containers

121

Anchor Peers

16

Leader Peers

16

CouchDB Instances

32

Orderers

8

Zookeepers

8

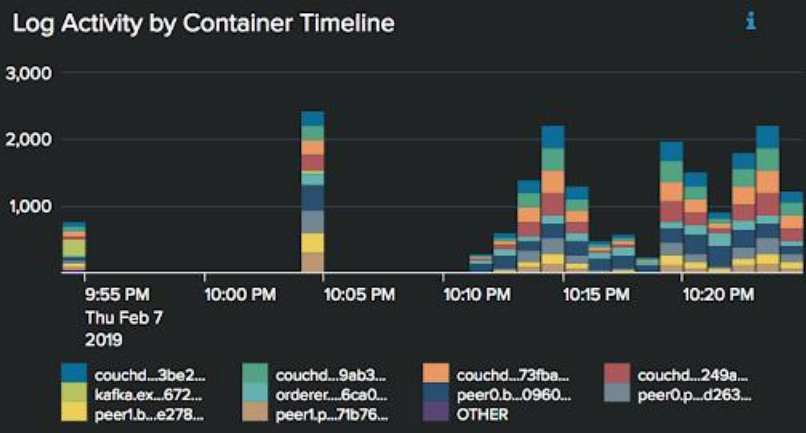
Kafka Nodes

10

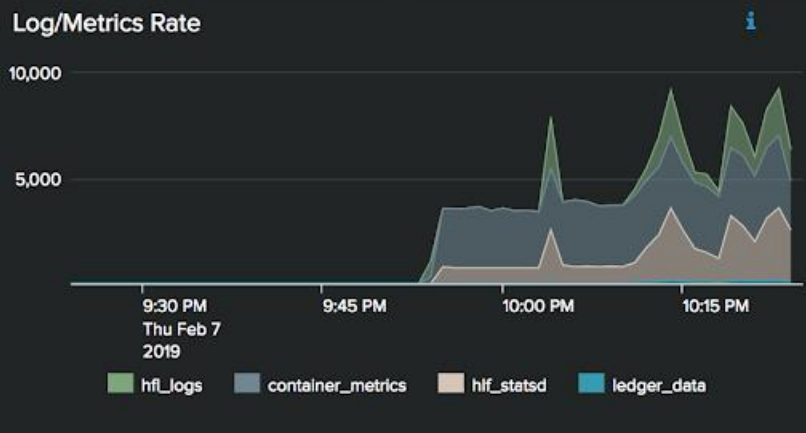
Certificate Authorities

5

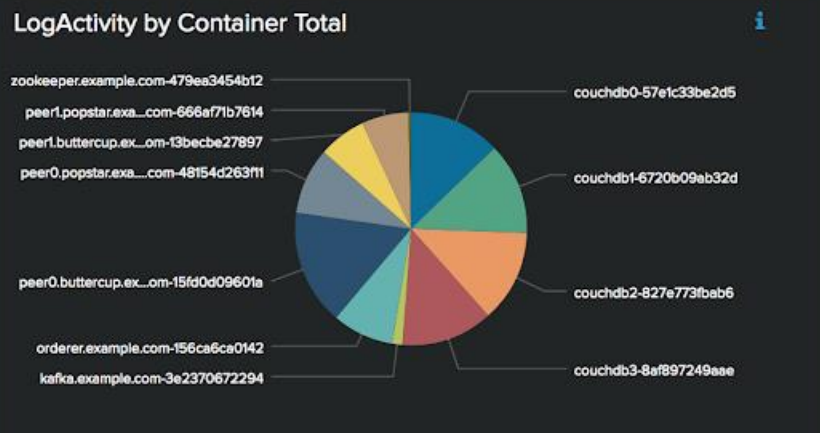
Log Activity by Container Timeline



Log/Metrics Rate



LogActivity by Container Total



Channels

Channel	Peers_Used	Activity	Block_Height
buttercup-go	peer0.buttercup.example.com-15fd0d09601a peer0.popstar.example.com-48154d263f11 peer1.buttercup.example.com-13becbe27897 peer1.popstar.example.com-666af71b7614		74
chaos-oath	peer0.buttercup.example.com-15fd0d09601a peer0.popstar.example.com-48154d263f11 peer1.buttercup.example.com-13becbe27897 peer1.popstar.example.com-666af71b7614		47

Step 8: Manage and Monitor

Summary

