Adopting Blockchain for enterprise asset management (EAM)

An IBM team discusses the challenges, architectural decisions, and lessons learned in their innovative solution

Mohan Venkataraman  
Murali Vridhachalam  
Alex Rosen  
Brian Arthur

Blockchain is a tremendously promising technology for transforming supply chain management to make it more efficient and less costly. This article describes the journey of an IBM team tasked with improving the end-to-end management of enterprise assets. We describe our starting point, the key architecture questions we had to answer, and our ultimate blockchain-based solution for our business. We share our lessons learned and thoughts on how to scale the system for future growth. Finally, we detail the IBM offerings we used and that are available to you for building your state-of-the-art blockchain solution.

Blockchain is a tremendously promising technology for transforming supply chain management to make it more efficient and less costly. This article tracks the journey of an IBM team tasked with improving the end-to-end management of enterprise assets. We describe our starting point, the key architecture questions we had to answer, and our ultimate blockchain-based solution for our business. We share our lessons learned and thoughts on how to scale the system for future growth. Finally, we detail the IBM offerings we used and that are available to you for building your state-of-the-art blockchain solution.
The impact of siloed systems and data on supply chains

Learn more about securely transferring assets in a business network:

To view this video, Applying Blockchain Technology to Asset Management, please access the online version of the article. If this article is in the developerWorks archives, the video is no longer accessible.

As one of the world’s largest providers of IT infrastructure and services, IBM manages tens of thousands of hardware and software assets in hundreds of IBM and customer facilities. These assets are manufactured at IBM and supplier plants, brought to IBM for configuration, and then deployed in customer data centers.

Tracking these assets as they move from manufacturing to deployment and eventually disposal is complex. Systems operated by IBM, suppliers, logistics companies, and customers are used to support this end-to-end process, with each system focusing on a particular function like planning, logistics, finance, warranty, and operations.

When everything goes perfectly, IBM’s assets move through the supply chain and the various systems quite efficiently. However, when issues arise, like an asset getting damaged in transit, or an incorrect item shipped as part of an order, getting all the systems to agree on the actual state of the asset is difficult. The systems and data are siloed, each with their own view of the asset. Resolving these issues is costly and creates delays that impact customer satisfaction and revenue recognition.

Introducing blockchain and our approach

In late 2016, IBM recognized an opportunity to improve its asset management system through the use of blockchain technology. A blockchain would capture all transactions and state changes that occur as an asset moves from manufacturing to deployment, including those actions that occur outside of IBM’s systems. This blockchain would provide a single source of truth for core information about each asset. This way, no matter what happened to an asset as it moved through the supply chain, all stakeholders and systems would know its status.

“We chose a Minimum Viable Product (MVP) approach to implement end-to-end asset management using blockchain technology. The MVP gave us a tangible solution within a limited scope.”

Blockchain Developer Center

Just getting started or already writing smart contracts and blockchain apps? Regardless of your starting point, sharpen your blockchain skills with developerWorks tutorials, courses, blogs, and community support in the Blockchain Developer Center.

Blockchain technology enables businesses to form a network that coordinates sharing of one or more ledgers. These shared ledgers differ from traditional databases in a few ways. Most
importantly, the ledgers are decentralized. Rather than being controlled by a single administrator or company, the ledger is controlled by the parties that share the ledger. Blockchain technology has four fundamental elements: a shared ledger, peer consensus, smart contracts (business rules), and privacy. Learn more in Blockchain basics: Introduction to distributed ledgers.

**Challenges in IBM's asset management process**

Blockchain inspired us to rethink, and enabled us to re-implement, how we work across enterprises. That said, since blockchain technology is new in the enterprise, we wanted to start by having our asset management blockchain shadow the existing process. Even without re-engineering the process, we found that we could use blockchain to augment our existing technology in ways that were valuable to the business.

IBM manages many types of its own assets and it also manages assets for its customers. We focused on a limited set of hardware assets for this project. Specifically, we focused on servers that would be installed in a data center.

Figure 1 represents the complete asset management lifecycle solution. For our purposes here, we focused on two key sub-processes: **Inventory capture & logistics**, and **Hardware tracking and financial management**.

**Figure 1. IBM's asset management process**

IBM's asset management team tracks assets as they move through a supply chain. Tracking starts when the asset is ordered, through productive use, until it is disposed (sold, recycled, etc.). Additionally, IBM's asset management services team performs software-related activities, including license tracking, compliance, and optimization. For MVP purposes, we started at the point when a serial number is generated for an individual asset by the manufacturer (serialization), proceeded to the recording of the asset in a financial ledger as a capital item (capitalization), which happens upon shipment, and continued to the receipt of an asset by IBM (receiving), including the errors that are identified in this step. The last event we included in our tracking was the installation of the asset in a data center.

The status of an asset as it moves through serialization, capitalization, receiving, and installation is recorded in different systems. It is not uncommon for an issue to arise that require the asset
management team to gather information from multiple systems in order to resolve it. In some cases, such as when an order received by IBM includes one or more assets with serial numbers that differ from those listed on an invoice, the issue requires IBM and the manufacturer to work together to resolve the issue.

The blockchain solution we built captures information about the asset as it moves from party to party in the supply chain (from manufacturer to IBM) and as the state of the asset is modified (for example, capitalized). Capturing this information in the blockchain gives IBM and its supply chain partners a single source of truth with regard to core asset information. That alone provides value since stakeholders have one system they can use to get what they need to know about an asset.

In the future, the value delivered by this blockchain will be substantially increased as we refactor existing systems to rely on the blockchain as their system of record. Even further gains will be achieved as we begin to use the blockchain as the information backbone of a more automated and streamlined process that reduces cycle times and errors.

**Using Blockchain to re-envision asset management**

The management of hardware and software assets has become a complex process within distributed environments. We often find that clients seek out third-party tools and services to obtain better control of their IT environment and to improve the inventory compliance posture in their organizations. These factors, coupled with the proliferation (and redundancy) of IT data repositories and system designs/process flows, make the most simplistic definition of "tracking an asset" complex, prone to inaccuracy, and with long delays in accessing critical information needed by service actors. The thought of secure data sitting in a highly restricted repository is now changing to cloud-based applications with aggregation and cognition of related metadata to bring renewed trust in data. Yes, automation strategies have addressed internal, back-office functions, but rarely do they give direct benefit to clients.

Our vision is a blockchain-based system to replace many of the legacy processes, tasks, and manual actions taken by various service delivery actors throughout the process. The system will replace inconsistent and out-of-sync data with a "block" of information that serves as the ultimate and immutable system of record truth, available in near real-time for all actors to use for ongoing information and transactions. Partnering with additional suppliers and clients is equally important to ensure the system is built to client needs and expectations.

Imagine a whole new delivery process for tracking assets. By using the business network offered in blockchain technology, all parties involved in a transaction on any key data element will be able to avoid disputes, have access to the single truth of status in near real-time, and mitigate, if not eliminate, software compliance exposures and even audits. We are moving away from reactive, mid-value service via administration and analytics to high-value, automated services that integrate all the necessary parties transacting on data with proactive checks in a new process that eliminates any redundancy, validations, reconciliations, and compliance documentation.

As shown in Figure 2, our vision is to change the technology and process for managing transactional updates applied to hardware, software, and associated contracts via blockchain.
technology, yielding a whole new system-of-engagement approach to all service delivery actions and clients, apart from the lifecycle of these hardware and software assets.

**Figure 2. Basic application of key blockchain concepts for asset management services**

![Diagram showing basic application of key blockchain concepts for asset management services]

**Key decisions in architecting the Blockchain solution**

We kicked off the MVP with the goal of examining whether we could manage the lifecycle of a marketable product such as a server. We applied the “IBM Design Thinking” approach, and limited the scope to what we could achieve in a 12-week period:

- Identify the segment of the business process with the pain points around data quality in the asset’s lifecycle
- Identify the personas involved and their specific issues to address
- Identify the business value realized by the solution

**Figure 3. Transactions and events**

![Diagram showing transactions and events]

Before making some key architectural decisions, we had to do some analysis:

- What are the technical options available to solve this?
- Who are the stakeholders, and how will they benefit from this?
- What are the stakeholders’ roles and responsibilities within their organizations?
- What is the common data of interest to all institutions or organizations/stakeholders?
• What are the inputs/outputs, and which applications produce and/or consume them?

We determined that applying blockchain technology will yield significant long-term benefits.

As a result, the key architectural decisions were:

• **Blockchain ledger as the system of record**: The blockchain will act as a single source of truth for an IBM-manufactured sellable asset such as a server, automatically tracking every change to its lifecycle, so it is auditable, trustworthy, and transparent.

• **Permissioned**: The participating organizations, such as manufacturing, logistics, finance, and the customer will be known and permissioned onto the blockchain network.

• **Shared data model**: A solution-specific data model that is of common interest to the participating organizations will be maintained on the blockchain.

• **Smart contracts**: Business rules governing the relationship between personas, transactions, and events affecting the asset will be maintained on the blockchain. For example, a shipper will not be able to flag a serial number mismatch, but a receiver can.

• **Proof-of-existence**: Proof received for transactions and events such as a "plant order" or "bill of lading" will be digitally hashed and recorded.

• **Client Interface**: The blockchain will subscribe to external transactions and events such as "asset shipped" or "asset installed" via a client interface.

### Solution architecture using Blockchain

The intent of the MVP was to build a limited-use, "production-ready" solution and prove the capability of the blockchain to help with asset lifecycle management issues. Hence, the team picked a set of core user stories that demonstrate business value and overcome technical challenges.

The solution architecture addresses the following user stories:

• Record the manufacturing and serialization of an asset through a transaction file upload of a manufacturing transaction via a user interface (temporary) provided in JSON format

• Record the manufacturing and serialization of an asset by receiving a JSON-formatted transaction file via an integration interface from manufacturing

• Record an event associated with the asset, such as shipment, capitalization, receipt, installation, and warranty

• Determine if the asset can be capitalized when a shipment event is received by interacting with the GFA system

• Encrypt and store the original transaction or event details and record the hash of those proofs on the blockchain

• Provide the ability to view the asset lifecycle including the proofs via a user interface using various methods — by asset serial number, by order identifier, and by shipment identifier

• Allow the whole application to behave according to the user role and permissions

We deployed the solution on an instance of the blockchain running on the public IBM Cloud. The access to the blockchain is provided by the SDK for Node.js client interface. All partner, application, and user interactions flow through the API or the user interface that sits on top of
it. The data that flows in and out of the blockchain are in JSON format, thus making it easy
to understand, consume, and process. In the MVP, the Node.js client integrates with the
manufacturing systems using the MQHub service available on IBM Cloud.

During the solution development process, we had to carefully avoid the pitfall of designing the
blockchain as "yet another application" or "a centralized operational database."

**Figure 4. Solution architecture**

![Solution Architecture Diagram](image)

**Table 1. Legend for Figure 4**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The Fabric — the physical or virtual servers running the software (one more more peers in a peer-to-peer network, and the Certificate Authority (CA) with or without security, privacy, and TSL enablement.)</td>
</tr>
<tr>
<td>1</td>
<td>Peer. Software running on a node on the network. Each peer works cohesively with other peers in the network and can perform one of many roles: non-validating, validating, endorser, committer.</td>
</tr>
<tr>
<td>2</td>
<td>CA, or Certificate Authority. Also known as Member services. Issues eCert (member enrollment certificates) and tCert (transaction certificates). Member organizations are enabled through the membersrvc.yaml file.</td>
</tr>
<tr>
<td>3</td>
<td>Ledger. An instance of the database associated with a peer. Every peer holds a copy of the ledger and records not only the application-specific data but also the world state resulting from every execution of the smart contract.</td>
</tr>
<tr>
<td>4</td>
<td>Node application components and Node SDK client that connects to the Fabric. The application components perform functions to receive and parse JSON files, emit events back to the supply chain, encrypt and decrypt files, manage sessions, and interface with the Node SDK client API. We have also provided a local client database for caching client specific data.</td>
</tr>
<tr>
<td>5</td>
<td>Browser-based user interface. Enables users to interact with the Asset Management application. Results to user requests may be drawn from the blockchain or from local application components and other external supply chain applications.</td>
</tr>
<tr>
<td>6 and 7</td>
<td>Message Hub: Send and Receive end-points: In the MVP, the send is on the side of the enterprise supply chain applications to send JSON structure events and transactions.</td>
</tr>
<tr>
<td>8</td>
<td>Enterprise supply chain applications, such as SAP, Manufacturing, and GFA, proofs-of-transaction events.</td>
</tr>
</tbody>
</table>
An encrypted document store. Saves encrypted copies of received proofs-of-transaction events and documents.

Local Fabric client-specific data cache

The MVP has to support internal users and business partners around the world, so it requires a highly available infrastructure. Our initial deployment is on a public IBM Cloud environment, which provides IBM Blockchain and Node JS as services. The deployment architecture is shown in Figure 5.

Figure 5. Deployment architecture

As shown in Figure 5, the environment within IBM is divided into various security zones. The Asset Management Application has been deployed on the "Red Zone" which is public. The supply chain applications are on the intranet or "Blue Zone". The final target production environment will be on HSBN (Highly Secure Business Network) once Fabric v0.6 becomes available.

Designing for scalability

Blockchain technology has the potential to address the interoperability challenges currently present in supply chain IT systems and to be the technical standard that enables customers, suppliers, logistics operators, financial institutions, contract manufacturers, and third-party service providers to securely share asset information.

Thus, the design we propose must be able to scale to support new asset types, events, business entities, and applications. During the MVP phase, we realized early on that while maintaining the immutability of the ledger, the solution must be scalable, extensible, and able to accommodate forward and backward compatibility with new releases of the fabric.

The interim reference architecture in Figure 6 shows our strategic thinking.
In Figure 6, the right two columns represent the blockchain cluster and the Fabric-Client clients accessing those chains. Each blockchain in the cluster provides different services to each other and the clients that connect to them. The Fabric-Client clients have been modeled to access a local database to be used by that instance of the client alone for client-specific data that requires caching. The clients also leverage a shared cloud database between them. In future architectures, a cloud database such as Cloudant would act as a shared data store for encrypted documents, instead of the file share being used in the MVP.

The middle column represents application UI and its components. Any number of them can connect to the Fabric-Client APIs and interact with the blockchain(s).

**Develop blockchain apps fast with Fabric Composer**

Fabric Composer (for Hyperledger Fabric v0.6 at the time of this writing) is a set of APIs, a modeling language, and a programming model to help you quickly define and deploy blockchain networks and applications.

*Overview | Quickstart | User story*

The second column from the left provides integration services such as MQ, ESB, and microservices. For example, the Fabric-Client client might invoke a GFA-Microservice to make a real-time Asset Capitalization check with the GFA system while processing a Shipment event. The very first column represents the various supply chain organizations including partners, and the applications that deliver various supply chain services. They communicate with the blockchain via the integration services or the UI or both.

After the initial MVP release, we positioned the solution to be able to accommodate and scale with future releases of the fabric. The chaincode (smart contract) can consume an event-state
configuration definition (shown in Listing 1) for a specific product category and execute upon it. This model enables adding new events and states that may be discovered in the future.

**Listing 1. Event-state configuration definition**

```json
Product_category {
  "EventStates": [  
    {  
      "EventID": "MANUFACTURING",
      "EventNO": 1,
      "RecType": "EventState",
      "EventDesc": "The Asset is manufactured & serialized",
      "ParentStates": [ "INIT"],
      "ChildStates": [ "SERIALIZED", "SHIPPED"],
      "Action": "AssetManufacturedEvent",
      "Documents": ["PlantOrder", "CustomerOrder"],
      "Signatures": ["Asset Admin", "Customer", "Order Manager"],
      "PreEvents": ["None"],
      "NextEvents": ["None"],
      "RoleAccess": ["Admin", "Manufacturer"],
      "TestEvent": "TestEventIntegrity"
    },
    {  
      "EventID": "SERIALIZED",
      "EventNO": 2,
    }
  ]
}
```

The smart contract validates the event against the current state of the asset, and applies the **Action**(s), verifies **signature**(s) and supporting **Document**(s), provided the client has the specific **RoleAccess**.

When a new event is added to the configuration, the chaincode provides a function to reinitialize its map that can be invoked by a client that has the permission and authorization(s). Obviously, exceptions to normal contracts may have to be coded into the chaincode, and a new version will have to be released.

The asset management data model, shown in Figure 7, is an integral part of the scalable design. The asset is treated as a first-class object whose state is influenced by events. Events are triggered by transactions originating within the supply chain environment.
The asset data model consists of the following attributes:

- **Asset**: A physical end-user product manufactured by IBM or its partners for internal consumption by an IBM end-user.
- **Asset ID**: A unique identifier, such as a Product Serial Number, Model Number, an RFID tag, a cryptographic address, or a combination of all.
- **Asset Metadata**: Properties that describe the asset such as Product Code, Model Number.
- **Transaction**: A supply chain contract between two parties such as a plant order, customer invoice, delivery order, and ASN.
- **Event**: Actions associated with transactions, such as an item delivered that will trigger invoicing.
- **Asset state**: Current context of the asset in its lifecycle, such as created, shipped, in-transit, received, installed, in-warranty, etc.
- **Asset Lifecycle**: A time-sequenced collection of asset states.

**Lessons learned**

We learned a few lessons while building this MVP, especially since the solution represents a paradigm shift in how we think about business, disruptive technology, and legacy systems.

- **Preparation**: Many of our team members were not experienced in blockchain, and in many cases, we noticed that continued exposure to legacy systems clouds thinking to focus narrowly on tactical issues.
  - The team must have at least one member who understands the new technology and how to adopt and embrace it.
  - The team must include SMEs who understand the business domain and have transformational thinking.
• **Analysis**: Our MVP kicked off with a one-day workshop to define the scope. It was followed by six iterations, or sprints.
  - A reasonably good analysis of the user stories, technical stories, and technical debt must be done prior to getting into the sprints. We recommend a Discovery phase, or Iteration 0, before each iteration or implementation.
  - Have sessions with users to understand how they will query information.

• **Designing**: While designing the application, keep in mind the principles of designing blockchain solutions and the target business value to be realized.
  - Information, or data, modeling is essential. Identify blockchain data, including shared blockchain client data that can live outside, as well as client-specific local data.
  - Identify the source systems that will provide the transactions and events, identify data attributes, and finalize the formats of the data.
  - Ensure real data is available for use in the MVP, especially if access is not provided to the production systems during the MVP phase.
  - Publish the signatures to blockchain APIs early on, so that client and server-side implementation can happen in parallel.

• **Building the solution**
  - Use a shared repository like GitHub. This helps with continuous integration.
  - Establish the environment with the correct versions of all the software in the ecosystem. These include Vagrant, Docker, Hyperledger Fabric, Git, HFC, Node.js, NPM, and Golang, to name a few.
  - Test the strategy for deploying to IBM Cloud. This includes configuring node services, blockchain services, and Message Hub services.

**Next steps**

From startups to enterprises, from paper-based businesses to technology companies, businesses can use flexible platforms and infrastructure offerings to design, deploy, and manage blockchain networks using IBM Blockchain offerings.

IBM Blockchain networks are built to benefit from decentralized control, but some cloud environments are open to vulnerabilities. Working with teams of security experts, cryptographers, hardware experts, and researchers, IBM has created essential cloud services for tamper-resistant, trusted blockchain networks.

**High Security Business Network: An isolated environment for business networks**

This Blockchain network plan runs in a secure infrastructure. This plan offers high levels of security that close any back doors to unauthorized access and tampering. Learn more at IBM Blockchain offerings and sign up for the High Security Business Network (HSBN) plan.

This plan provides easy-to-use, built-in workflow tools, that allow members of a blockchain network to scale and grow as multi-company networks grow. This plan also provides backup and restore procedures, logging and monitoring, rigorous security, and a high-availability/fault-tolerant architecture that eliminates single points of failure.
Fabric Composer: An open-source tool for rapid blockchain application development on the HyperLedger Fabric

A quick way to develop applications on the Fabric is to use the new open-source tool, Fabric Composer (currently based on Hyperledger Fabric v0.6, but updates to reflect v1.0 will be available soon). Fabric Composer abstracts the details from Hyperledger Fabric and makes it easy to:

- Build and test a blockchain business network
- Develop applications to interact with your network
- Integrate existing systems with your network

Get an overview of Fabric Composer, and follow the Quickstart. When you complete the Quickstart, you'll have a running local instance of Hyperledger Fabric and a deployed business network, ready for app development.

About the authors

Mohan Venkataraman is the CTO and Principal Consultant at IT People Corporation.

Murali Vridhachalam is an IBM Distinguished Engineer in the IBM Analytics Office and IBM Data Office.

Alex Rosen is the VP of Business Development for Blockchain Consulting Services at IT People Corporation.

Brian Arthur is an IBM Client Analytics Transformation and Blockchain Program Project Manager
Related topics

- Blockchain Developer Center
- Hyperledger Fabric
- Hyperledger community
- Hyperledger discussion channels
- IBM Blockchain basics: Introduction to distributed ledgers
- IBM Blockchain 101: Quick-start guide for developers
- IBM Blockchain courses for developers (free)
- IBM Blockchain videos on developerWorks TV

© Copyright IBM Corporation 2017
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)