Paper updates
https://docs.google.com/document/d/1Z4M_qwILLRehPbVRUsJ3OF8Iir-gqS-ZYe7W-LE9gnE/edit#heading=h.m6iml6hqrnm2

Hyperledger Whitepaper

Abstract

This paper describes industry use cases that drive the principles behind a new blockchain fabric, and outlines the basic requirements and high-level architecture based on those use cases. The design presented here describes this evolving blockchain fabric, called **Hyperledger**, as a protocol for business-to-business and business-to-customer transactions. Hyperledger allows for compliance with regulations, while supporting the varied requirements that arise when competing businesses work together on the same network. The central elements of this specification (described below) are smart contracts (a.k.a. chaincode), digital assets, record repositories, a decentralized consensus-based network, and cryptographic security. To these blockchain staples, industry requirements for performance, verified identities, private and confidential transactions, and a pluggable consensus model have been added.

For questions regarding Hyperledger terminology, check out our glossary.

Background

Blockchain is an emerging technology pattern that can radically improve banking, supply-chain, and other transaction networks, creating new opportunities for innovation and growth while reducing the cost and risk of related business operations. With the rapid emergence of Bitcoin in the transactions domain since 2009, many businesses and industries have invested significant resources in investigating the underlying technology that powers the popular, yet controversial, cryptocurrency.

Blockchain is a peer-to-peer distributed ledger technology that first gained traction in the financial industry because of its capacity to issue, trade, manage, and service assets efficiently and securely. The distributed ledger makes it easy to create cost-efficient business networks without requiring a central point of control, in marked contrast to the world of SoR (System of Records), where every member in the ecosystem needs to maintain its own ledger system and reconcile transaction updates with one another in inefficient, expensive, and often non-standardized inter-organizational operation flows.

As the shared ledger concept gains tracking in the business world, blockchain **smart contracts** are also getting a lot of attention from industry [Eth]. A smart contract is a collection of business rules which are deployed on a blockchain and shared and validated collectively by a group of stakeholders. A smart contract can be very useful in automating business processes in a trusted way by allowing all stakeholders to process and validate contractual rules as a group. In Hyperledger, smart contracts are implemented by chaincode.

Bitcoin and other cryptocurrencies were designed to be completely open, decentralized, and permissionless: anyone can participate without establishing an identity; one only has to contribute by spending computation cycles. Under the Bitcoin model of blockchain, there is no central authority that controls admission; these networks have

been called permissionless. Through their nature of requiring innumerable proof-of-work computations, they are costly to operate [N09].

Hyperledger takes a novel approach to the traditional blockchain model, in part by managing the admission of participants at its core. In other words, Hyperledger is a permissioned, shared ledger. Hyperledger saves computation cycles, scales well, and responds to the multitude of industrial use case requirements by providing a secure, robust model for identity, auditability and privacy.

Entering 2016, blockchain awareness has reached the point where demand for an industrial-strength, extensible solution is surging.

Why a new fabric

As a fledgling technology, blockchain has fallen short of meeting the multitude of requirements inherent in the complex world of business transactions. Scalability challenges, and the lack of support for confidential and private transactions, among other limitations, make its use unworkable for many business-critical applications. To meet the varied demands of the modern marketplace, Hyperledger has been designed for a broad array of industry-focused use cases, thereby extending the work of the pioneers in the field by addressing the existing shortcomings.

Our vision

We have developed a vision for how blockchain technology will evolve and change the fundamentals of modern commercial transactions. Based on this vision, we have

examined industry use cases, identified key requirements, and designed and built a system that we believe will bring blockchain technology to the masses.

Note: To prepare you for the material below, we strongly recommend reviewing our glossary first.

A world of many networks

Hyperledger is based on the expectation that there will be many blockchain networks, with each network ledger serving a different goal. While there may be a popular single instance of a general-use network, there is no requirement for any one network ledger to rely upon any other network for its core functionality. Despite this level of network independence, Hyperledger still requires an addressing system that allows transactions on one ledger to discover and utilize appropriate transactions and smart contracts (chaincode) on other ledgers.

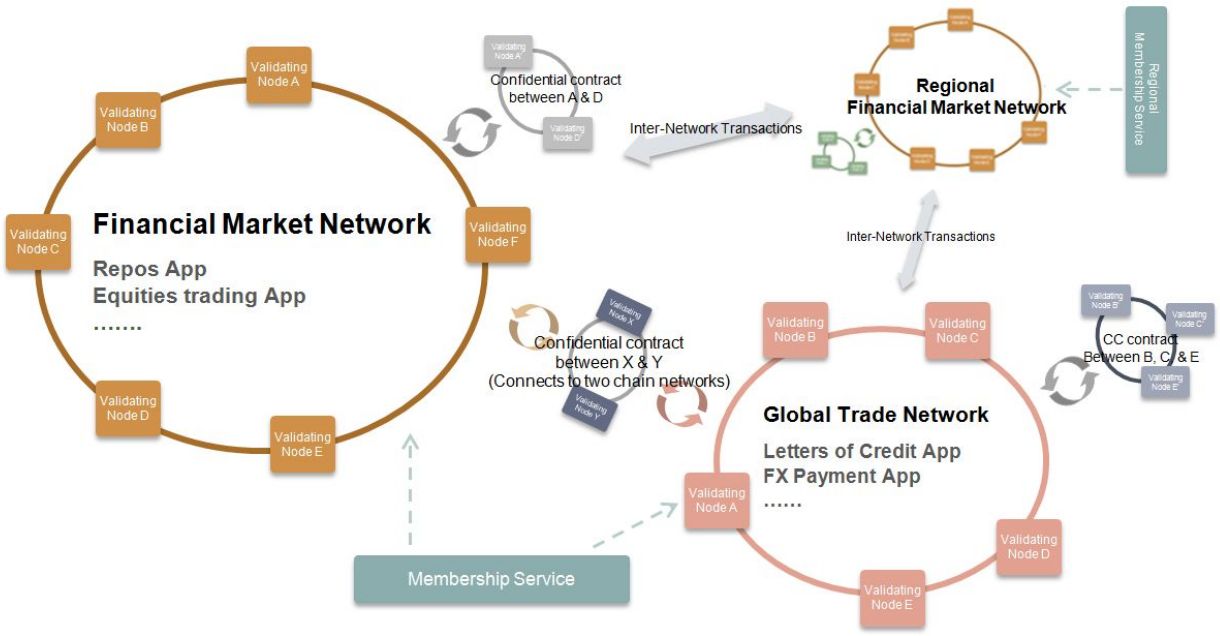


Figure 1: A world of many blockchain networks

Increasing demand for permissioned networks

We describe permissioned networks as those where validating and non-validating nodes are run by known whitelisted organizations, and where transactors on the network are granted an identity from an issuing authority service on the network. Depending on the purpose of the network, the issuing authority assigns the appropriate level of access that is required to obtain an identity and transact on the network. A network can be run very publicly, making it easy to integrate into a mobile app project, or it can be completely private and known only to invited participants whose identities have been validated. Because the Hyperledger fabric is designed to support many networks with many different purposes, and to allow addressing between them, the protocol must allow for different types of uses and distinct levels of permissioning.

Importance of both privacy and confidentiality

We believe that one of the fundamental requirements for any blockchain fabric is that the identity and patterns of behavior of any party on a network must be impossible for unauthorized parties to ascertain by inspecting the ledger. We also anticipate a requirement to allow blockchain users to make certain business logics and/or other parameters of a transaction confidential, rendering them inaccessible to anyone other than the stakeholders for the contract or the asset being transferred.

Industry use cases

We have compiled a set of initial blockchain requirements that are considered essential for supporting the following abstract use cases.

(Note: The use cases here help guide architecture and test-driven development. While still a work in progress, the use cases should be something all contributors agree on: both in the content and the stack-ranked prioritization of them. Propose changes if you feel these miss the mark. It is ideal if there are no more than four abstract use cases, and three is preferred.)

Business contracts

Business contracts can be codified to allow two or more parties to automate contractual agreements in a trusted way. Although information on blockchain is inherently public, B2B contracts often require privacy control to protect sensitive business information from being disclosed to outside parties that also have access to the ledger.

While confidential agreements are a key business case, there are many scenarios where contracts can and should be easily discoverable by all parties on a ledger: for example, a ledger used to create offers (asks) seeking bids. This type of contract might need to be standardized so that bidders can easily find them.

Asset depository

Assets such as financial securities must be able to be dematerialized on a blockchain network, so that all stakeholders of an asset type will have direct access to each asset, allowing them to initiate trades and acquire information on an asset without going through layers of intermediaries. Trades should be settled in near real time, and all stakeholders must be able to access asset information in near real time. A stakeholder should be able to add business rules for any given asset type, which further reduces operating costs by implementing automation logic. The creator of the asset must be able to make the asset and any rules associated with the trading of that asset private and confidential, or public as the use case warrants.

Supply chain

The blockchain fabric must provide a means to allow every participant on a supply chain network to input and track sourcing of raw materials, record parts manufacturing telemetry, track provenance of goods through shipping, and maintain immutable records of all aspects of the production and storage of a finished good through to sale and afterwards. In addition to employing both the **Business contracts** and **Asset depository** patterns described previously, this case emphasizes the need to provide deep searchability, backwards in time through many transaction layers. This requirement is at the core of establishing provenance for any manufactured good that is built from other component goods.

For more details about use cases and their requirements, and to visualize how these use cases can be plugged into a blockchain-based system, please click here.

Featured requirements

The featured requirements (described below) are based on robust industry use cases, which have been driven into the resulting Hyperledger architecture. These requirements include identity and auditability, private transactions, confidential contracts, modular consensus, performance, scalability, chaincode and smart contracts.

Identity and auditability

Although private transactions are important, business usage of blockchain also requires compliance with regulations and access for regulators to investigate transaction records. A party to a transaction must also be able to prove its identity and ownership of an asset after the fact (sometimes years after the fact), without the mechanism for

establishing that identity also enabling the determination of a party's identity and/or their activities on the ledger.

As a result, the Hyperledger protocol starts with a cryptographic certificate encapsulating a user’s confidential data, which is registered on a Registration Authority. The Registration Authority can issue and revoke identities that are participating in a network. From each identity, the protocol can generate security keys for members to transact on a network, which conceal the identities of the transacting parties, providing privacy support to the network.

Still have questions on identity and auditability? Check out the identity management section of FAQ.

Private transactions and confidential contracts

If transaction patterns are open to being observed and interpreted, shared ledgers could give away details about business relationships that should not be revealed to competitors. In tight supplier/buyer communities, even one party's relative volume of trade is information that should not be revealed by a system supporting trade between parties. Therefore, a business-ready blockchain must provide mechanisms to conceal identity, transaction patterns, and terms of confidential contracts from unauthorized third parties.

Within Hyperledger, content confidentiality is achieved by encrypting the transactions such that only the stakeholders can decrypt and execute them. In addition, a piece of business logic (realized by a smart contract) can also be cryptographically secured (if confidentiality is required by its stakeholders) so that it only gets loaded and decrypted at runtime. This is further explained in the architecture below.

Still have questions on confidentiality? Check out the confidentiality section of FAQ.

Modular consensus

Because different industries and regions may run their own networks, different networks might need to deploy different consensus algorithms to fit their usage scenarios. Consensus algorithms under the Hyperledger protocol must be pluggable, allowing users to select the consensus algorithm of their choice during deployment. The Hyperledger protocol will provide an implementation of Byzantine Fault Tolerance (BFT) in its initial release, using the PBFT protocol [CL02]. We anticipate that the community will contribute additional consensus algorithm modules in the future.

Still have questions on consensus and want to explore more about Hyperledger's pre-packaged consensus implementations? Check out the consensus section of FAQ.

Logic = Chaincode = Smart contracts

Blockchain logic, often referred to as "**smart contracts**," are self-executing agreements between parties that have all relevant covenants spelled out in code, are settled automatically, and can be dependent upon future signatures or trigger events. In the Hyperledger project, we call this "chaincode" to help establish clarity between blockchain logic and the human-written contracts that they can sometimes represent. (This term is still under review and may change.)

The chaincode concept is more general than the smart contract concept that was defined by Nick Szabo [SC]. Chaincode can be written in any mainstream programming language, and executed in containers inside the Hyperledger context layer. Chaincode provides the capability to define smart contract templating language (similar to Velocity or Jade), and to restrict the functionality of the execution environment and the degree of computing flexibility to satisfy the legal contractual requirements.

Still have questions on chaincode and how is it used to create business contracts and digital assets? Check out thechaincode section of FAQ.

Performance and Scalability

If blockchain becomes the fabric of an economically-aware Internet, then it must be designed for performance over the long term. A ledger or set of ledgers must be able to operate continuously for 100+ years, and still allow discoverability, search, identity resolution and other key functions in user-acceptable timeframes. Likewise, the number of nodes and transactors on a given network could become extremely large over time; the fabric must be able to handle such expansion without performance degradation.

Still have usage related questions? Check out the usage section of FAQ.

as indicated in Figure 1.

Architecture

Figure 2 below shows the Hyperledger reference architecture aligned in three categories: Membership, Blockchain and Chaincode. These categories are a logical structure, not a physical depiction of partitioning of components into separate processes, address spaces or (virtual) machines.

Some of these components will be built from the ground up, some will use existing open-source code, and some will interface with existing services to fulfill the required functions.

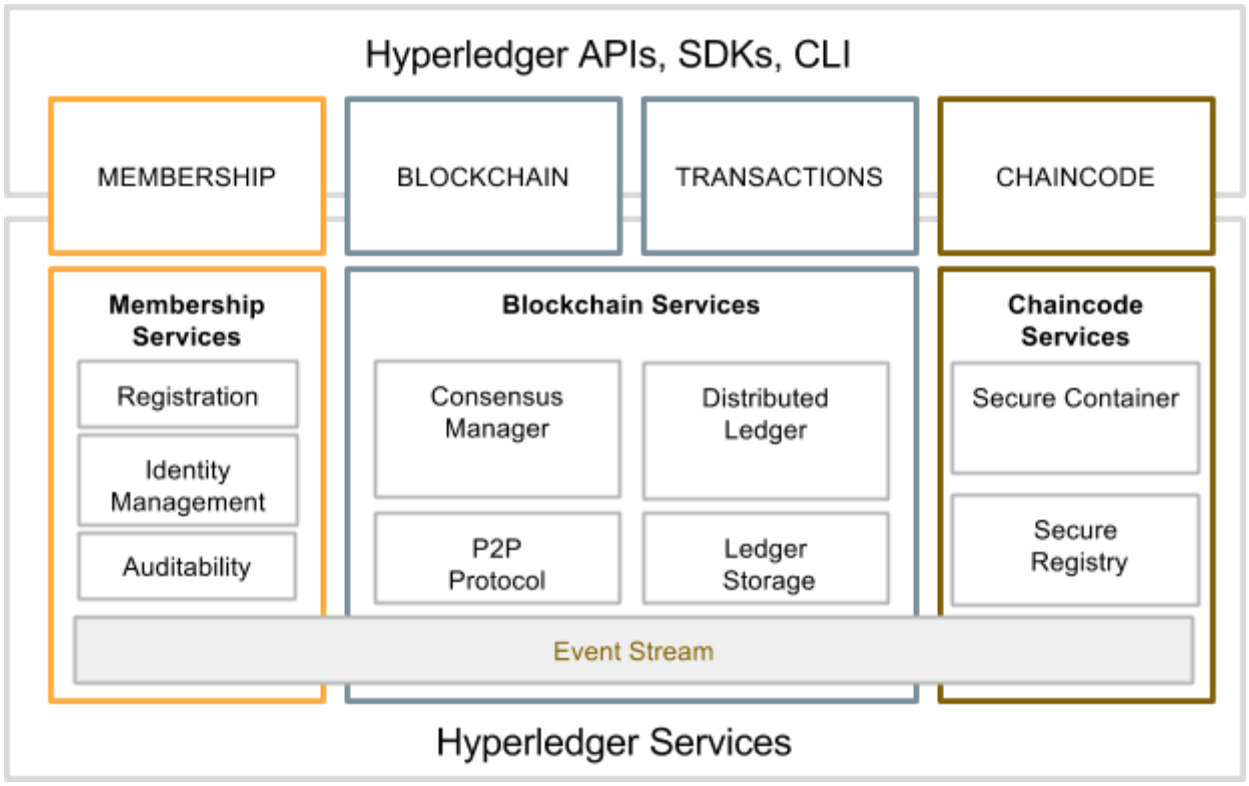


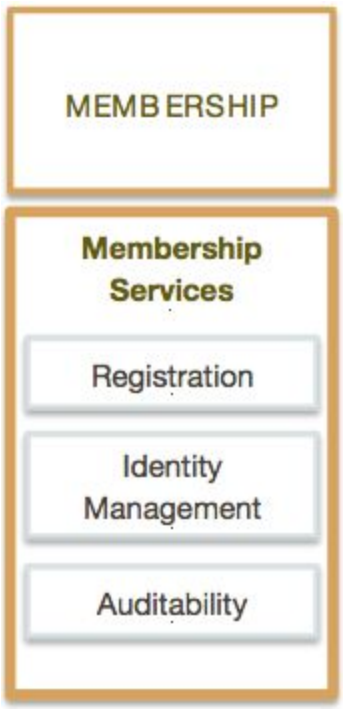
Figure 2: Hyperledger reference architecture

Membership services manages identity, privacy and confidentiality on the network. Participants register to obtain identities, which enables the Attribute Authority to issue security keys for transacting. Reputation Manager enables auditors to view transactions pertaining to a participant, assuming that each auditor has been granted proper access authority by the participants.

Blockchain services manage the distributed ledger through a peer-to-peer protocol, which is built on HTTP/2. The data structures are optimized to provide efficient schemes for maintaining the world state replicated at many participants. Different consensus algorithms guaranteeing strong consistency (tolerating misbehavior with BFT, tolerating delays and outages with crash-tolerance, or tolerating censorship with proof-of-work) may be plugged in and configured per deployment.

Chaincode services are a secured and lightweight way to sandbox the chaincode execution on validating nodes. The environment is a "locked down" and secured container with a set of signed base images which contain secure OS and chaincode language, runtime and SDK images for Golang (ready), Java (planned), and Node.js (planned). Additional programming languages can be enabled if required.

MEMBERSHIP



Hyperledger is a private-validator network protocol, so all entities are required to register with membership services to obtain an identity with access and transaction authority on the network. Validators during network setup can determine the level of permission that is required to transact. Network setup also defines the network as permissive, allowing ease of access and support for rapid and high adoption, or restrictive for a more controlled environment.

Registration service offers the control and management of authorizations for Hyperledger participation.

Identity Management service provides the management of assurance and authorized disclosure of association of identities and roles to Hyperledger participants.

Auditability service offers the capability to provide authorized entities with the means to link transactions of individual users, or groups of users according to

	affiliations or roles, and to access the activity of a particular user of the system, or the operation of the system itself.

BLOCKCHAIN

	Blockchain services consists of three key components: Peer-to-Peer (P2P) Protocol, Distributed Ledger and Consensus Manager. **P2P Protocol** uses Google RPC, which is implemented over HTTP/2 standards, providing many capabilities including bidirectional streaming, flow control, and multiplexing requests over a single connection. Most importantly, it works with existing Internet infrastructure, including firewalls, proxies and security. This component defines messages used by peer nodes, from point-to-point to multicast. **Distributed Ledger** manages the blockchain and the world state by implementing three key attributes:

	• Efficiently calculating a cryptographic hash of the entire dataset after each block. • Efficiently transmitting a minimal "delta" of changes to the dataset, when a peer is out of sync and needs to "catch up." • Minimizing the amount of stored data that is required for each peer to operate.

Distributed Ledger uses RocksDB to persist the dataset, and builds an internal data structure to represent the state that satisfies the three attributes. Large files (documents, etc.) are stored in off-chain storage, not on the Ledger. Their hashes can be stored on-chain as part of the transactions, which is required to maintain the integrity of files.

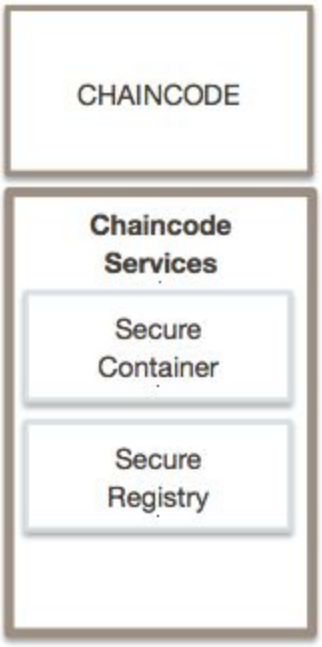
Hyperledger supports two types of transactions: code-deploying transactions and code-invoking transactions. A code-deploying transaction can submit, update or terminate a piece of chaincode, and the validating node must protect the authenticity and integrity of the code and its execution environment. By contrast, a code-invoking transaction is an API call to a chaincode function, which is similar to how URI invokes a servlet in JEE. Note that each chaincode maintains its own state, and a function call is a common method for triggering chaincode state changes.

Consensus Manager is an abstraction that defines the interface between the consensus algorithm and the other Hyperledger components. Consensus Manager receives transactions, and depending on the algorithm, decides how to organize and when to execute the transactions. Successful execution of transactions results in changes to the ledger.

Hyperledger provides an implementation of the Byzantine Agreement, with advanced features in fault tolerance and scalability.

Event Hub in a decentralized network is complex in nature, because an event can appear to occur multiple times, once on each peer node. Callbacks can end up receiving multiple invocations for the same event. Therefore, a peer node (preferably non-validating and local) manages the event pub/sub that applications are interacting with. The peer node emits events as conditions satisfied, in no particular order. Events do not persist, so applications should capture events if required.

CHAINCODE



As defined in the previous sections, a chaincode is a decentralized transactional program, running on the validating nodes.

Chaincode Services uses Docker to host the chaincode without relying on any particular virtual machine or computer language. Docker provides a secured, lightweight method to sandbox chaincode execution. The environment is a "locked down" and secured container, along with a set of of signed base images containing secure OS and chaincode language, runtime and SDK images for Golang. Additional programming languages can be enabled, if required.

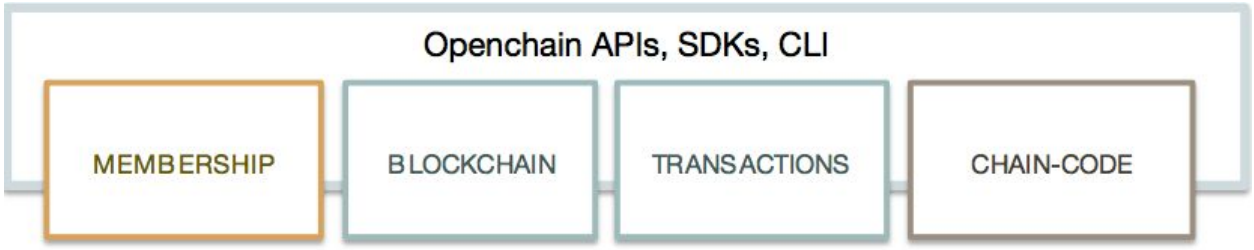
	Secure Registry Services enables Secured Docker Registry of base Hyperledger images and custom images containing chaincodes. The world state represents the state for every chaincode. Each chaincode is assigned its own state that can be used to store data in a key-value format, where keys and values are arbitrary byte arrays. The world state also contains the block number to which it corresponds.

Chaincode transactions are time bounded and configured during chaincode deployment, which is similar to a database call or a Web service invocation. If a transaction times out, it is considered an error and will not cause state changes on the Ledger. One chaincode function can call another chaincode function if the callee has the same restrictive confidentiality scope; that is, a confidential chaincode can call another confidential chaincode if they share the same group of validators.

As transactions are run in a new block, a delta from the world state in the last block on the blockchain is maintained. If consensus is reached for the current block, the changes are committed to the database, and the world state block number is incremented by 1. If peers do not reach consensus, the delta is discarded and the database is not modified.

Application programming interface

Hyperledger includes the REST and JSON RPC APIs, events, and an SDK for applications to communicate with the network. Typically, applications interact with a peer node, which requires some form of authentication to ensure that the entity has proper privilege; messages from a client are signed by the client identity and verified by the peer node.

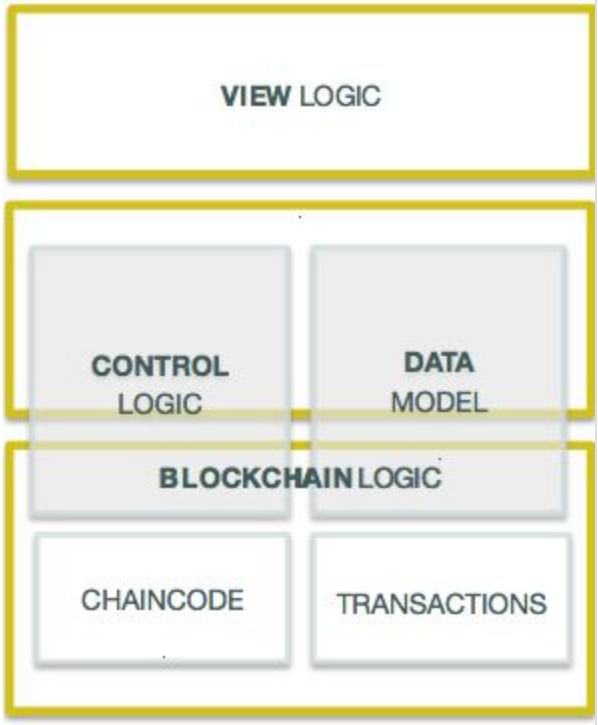


Hyperledger provides a set of CLIs to administer and manage the network. CLI can also be used during development to test chaincodes. REST API and SDK are built on top of JSON-RPC API, which is the most complete API layer. SDK will be available in Golang, JavaScript, and Java; additional programming languages can be added as necessary.

The API spans the following categories:

- Identity - Enrollment to get certificates or revoke a certificate
- Address - Target and source of a transaction
- Transaction - Unit of execution on the ledger
- Chaincode - Program running on the ledger
- Blockchain - Content of the ledger
- Network - Information about the blockchain network
- Storage - External store for files or documents
- Event - Sub/pub events on blockchain

Application model



An Hyperledger application follows a MVC-B architecture – Model, View, Control, Blockchain.

- VIEW LOGIC – Mobile or Web UI interacting with control logic.
- CONTROL LOGIC – Coordinates between UI, Data Model and Hyperledger APIs to drive transitions and chaincode.
- DATA MODEL – Application Data Model – manages off-chain data, including documents and large files.
- BLOCKCHAIN LOGIC – Blockchain logic are extensions of the Controller Logic and Data Model into the Blockchain realm. Controller logic is enhanced by chaincode, and the data model is enhanced with transactions on the blockchain.

For example, a PaaS application using Node.js might have a Web front-end user interface or a native mobile app with a backend model provided by a data management service. The control logic may interact with one or more

	chaincodes to process transactions on the blockchain.

Network Topology

There are three potential deployment models: Cloud hosted one network, Cloud hosted multiple networks, and Participant hosted intranet.

The simplest and most efficient topology is the **Cloud hosted one network**, where each participant owns a number of peer nodes, including validating nodes. Even though the network is in a cloud and hosted by a vendor who owns the physical hardware, the participants contractually control the computing resources, making the configuration decentralized within a centralized environment.

The **Cloud hosted multiple networks** environment allows participants to have their peer nodes hosted by any cloud provider, given that peer nodes can connect to one another over HTTPs.

The **Participant hosted intranet** environment uses networks that are owned by the participants, with an HTTPs channel.

Conclusion

Hyperledger's mission is to bring blockchain technology to mass markets. After reviewing the available blockchain solutions and hearing use cases from both industry

leaders and technology evangelists, we are convinced that blockchain will be an extremely important technology pattern that could revolutionize many industries and businesses.

We have observed that industry is urgently calling for a business-ready blockchain fabric that is both efficient and scalable, and offers enterprise-grade support for privacy and confidentiality. We have also discovered many different categories of use cases, and each may require a different underlying blockchain implementation.

To fully realize the potential of blockchain technology and to create a standard that can be adopted into many different uses, we designed the Hyperledger fabric to be both flexible and extensible. In addition, we are also leading the default implementation of the Hyperledger protocol, which includes many of the latest advances in the various computer science disciplines.

To enhance your understanding of the Hyperledger protocol, we invite you to read our protocol specification, which will prepare you to start using Hyperledger to build your applications, and to contribute to the project.

References

- [CL02] Miguel Castro and Barbara Liskov, Practical Byzantine Fault Tolerance
- [SC] Wikipedia, Smart Contract
- [N09] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System
- [Eth] Ethereum Whitepaper