Hyperledger Fabric V1
The Linux Foundation is the organization of choice for the world's top developers and companies to build ecosystems that accelerate open technology development and commercial adoption. Together with the worldwide open source community, it is solving the hardest technology problems by creating the largest shared technology investment in history. Founded in 2000, The Linux Foundation today provides tools, training and events to scale any open source project, which together deliver an economic impact not achievable by any one company.

The Linux Foundation has 16 years experience of providing governance structure and infrastructure to support the development of large scale, successful open source projects such as:

- Cloud Foundry
- Node.js
- Xen Project
Linux Foundation’s Hyperledger Project

– *Open Ledger Project* announced December 17, 2015 with 17 founders, now over 80 members

– *Hyperledger Project* rebrand in February 2016

– Collaborative effort to advance Blockchain technology by identifying and addressing important features for a cross-industry open standard for distributed ledgers that can transform the way business transactions are conducted globally

– Open source, open standards, open governance

Enable adoption of shared ledger technology at a pace and depth not achievable by any one company or industry

QUICK FACTS

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chairman</td>
<td>Blythe Masters/DAH</td>
</tr>
<tr>
<td>Executive Director</td>
<td>Brian Behlendorf</td>
</tr>
<tr>
<td>Technical Chair</td>
<td>Chris Ferris/IBM</td>
</tr>
</tbody>
</table>

Contribution: 44,000 lines of code in February 2016

Sprint to one codebase with unified thinking

Staged releases

www.Hyperledger.org
Hyperledger Fabric: Updated Roadmap and Releases

- Custom events
- Version indicator (log and cli)
- CC deploy SDK API

- Consensus 1
- Life-cycle SCC
- Error handling
- Tx simulation nw-set
- File-based datastore

- ACL
- Kafka 1
- Multichannel 1
- HSM support PKCS11

- Sec code hardening
- Bug fixes

- Member services 1
- Enhance Ledger API
- Status codes & msg’s
- Event listener SDK

- Enhance protocol
- Life-cycle SCC
- SDK specification
- SDK submitting TX

- Auditability API
- State cache
- Upgrade chaincode
- Kafka 2
- Multichannel 2

- Bug fixes

Skeleton drop - 11/11/16
Complete basic transaction flow of v1.0 architecture, from node.js SDK to commit on block.

Alpha drop of v1.0 - 12/17/16
Previous Alpha features move to Beta, additional features introduced as Alpha.

Beta driver of v1.0 – 1/31/17
Previous Alpha features move to Beta, additional features introduced as Alpha.

GA - 3/31/17
Overview of Hyperledger Fabric v1
Today’s V0.6 Hyperledger Fabric Blockchains

• In a PBFT blockchain network, smart contract (chaincode) is run on all validating nodes.
• If there are a large number of peers with update transactions being submitted, the network may not scale well.
• For example, take a network built around vehicle leasing in the UK:

- With PBFT, the DVLA, all the manufacturers, and all the leasing companies have to process every update transaction, if they all run their own validating nodes.
Considerations - Confidentiality

- In the vehicle leasing example, the manufacturers and leasing companies may not want their competitors to see details of transactions.
- For example, a lease company could use evidence of discount rates for other companies against the manufacturer in future negotiations.
- Encryption of data can help, but businesses are keen to avoid transactions being executed on competitors nodes.

![Diagram showing vehicle leasing companies and discounts](image-url)
Current Architecture (v0.6)

- **SDK**
  - keys

- **membership**
  - ECA, TCA, TLS-CA

- **peer**
  - Consensus
  - Ledger
  - Events
  - Chaincode
  - state

Connections:
- **SDK** to **membership** via enroll
- **SDK** to **peer** via transact
What We Have Learned

We need to:

• Support broader confidentiality
• Scale the number of participants and transaction throughput
• Eliminate non deterministic transactions
• Enable pluggable data store
• Be able to dynamically upgrade fabric and chaincode
• Remove SPF and enable multiple providers of Membership Services
Overview of Hyperledger Fabric v1

Example transaction flow
v1.0 Architecture

SDK

membership

No SPoF
No SPoT

peer

Endorser
Committer
Ledger
Events
Chaincode
state

Consensus Service

Order TXs in a batch according to consensus

enroll

Proposal

Submit Tx

Client App proposes a transaction for **Smart Contract A** to the Endorsing peer E₀.

**Note:** Endorsement policy: “E₀, E₁ and E₂ must sign”. E₃, E₄ and E₅ are not part of the policy.
A sample transaction (2/7)

Endorsing peer $E_0$ endorses a tx and (optionally) “anchors it” with respect to the ledger state version numbers. An “anchor” contains all data read and written by the contract that is to be confirmed by other endorsers.
The client requests further endorsement from E₁ and E₂. The client may decide to suggest an anchor obtained from E₀ to E₁ and E₂.
The Endorsing peers $E_1$ and $E_2$ send the endorsement to client
Client formats the transaction and broadcasts it to the consensus-service nodes for inclusion in the ledger.
The consensus service delivers the next block in the ledger with the consented transaction. $E_4$ and $E_5$ are not on the same channel and therefore do not receive an update.
The peers validate the block received from the consensus-service and update their ledger and world-state.
Raw and Validated Ledger (batches vs. blocks)

- Raw Ledger: may contain invalid transactions – output by consensus service
- Validated Ledger: a filtered version of Raw Ledger produced by Peers
Example PBFT network (i.e. as today)

- All peers connect to the same system channel (red).
- All peers have the same chaincode and maintain the same ledger.
- Endorsement by peers $E_0$, $E_1$, $E_2$ and $E_3$
• Peers $E_0$ and $E_3$ connect to the red channel for chaincode $A$ and $B$
• Peers $E_1$ and $E_2$ connect to the blue channel for chaincode $Y$ and $Z$
Pluggable world state
How is data managed on the ledger?
**World state today (v0.6)**

- Smart contracts can store data in the Blockchain using a key/value store.
- This data is managed and persisted using RocksDB.
  - RocksDB is an open source, embeddable key-value store.
- Underlying persistence store is hidden from developer by the chaincode APIs:
  - `GetState()` and `PutState()`
- World state is currently quite restrictive – for example, there is limited capability to run complex queries against stored data.
World-state in the future

• Underlying storage to be made “pluggable”, so network operators can choose which implementation to use.
• First implementation will use Apache CouchDB:
  – NoSQL database that stores JSON documents.
  – Extensive query capabilities.
Permissioned Ledger Access
Transaction and identity privacy
Requests certificates 1xEcert, NxTcert

Consensus Network

Blockchain

User A uses

Ecert

Tcert

Invokes SC txn (signed with TkeyA, encrypted with Vkey)

TkeyA

Vkey

Blockchain

User B uses

Accesses ledger

Enrollment certificates (Ecerts) and Transaction certificates (Tcerts) can only be linked by CA and user

Certificate Authority

Membership

Requests certificates 1xEcert, NxTcert (stored in wallet)

Application

deployed on every validating peer

Smart contract

(signed with Ekey of origin, encrypted with validators’ key)

Application

Accesses ledger

Membership

Enrollment certificates (Ecerts) and Transaction certificates (Tcerts) can only be linked by CA and user

Certificate Authority

Application

Accesses ledger

Membership

Requests certificates 1xEcert, NxTcert (stored in wallet)

Application

deployed on every validating peer

Smart contract

(signed with Ekey of origin, encrypted with validators’ key)
Transaction and Identity Privacy

• Transaction Certificates, Tcerts
  – Disposable certificates, typically used once, requested from Transaction CA
  – Tcert derived from long term identity - Enrollment Certificate, Ecert
  – Only Transaction CA can link Ecert and Tcert

• Permissioned Interactions
  – Consumer shares public Tcert to provider
  – Provider invokes chain code transaction as usual, but
    • Signs with provider’s private Tcert for authentication
    • Encrypts with provider and consumer Tcerts for subsequent access
  – Consumers can subsequently access ledger data using their private key

• Secure chain code
  – CC can also be signed and encrypted, to keep verify and secure contract details
  – Signing is by contract owner/author
  – Encryption ensures only validators can see and execute transaction chain code
Summary and Next Steps

For users
Summary

• V1 Hyperledger Fabric
  • Support broader confidentiality
  • Scale the number of participants and transaction throughput
  • Eliminate non deterministic transactions
  • Enable pluggable data store
  • Be able to dynamically upgrade fabric and chaincode
  • Remove SPF and enable multiple providers of Membership Services
Thank You!
Next Generation Consensus
Details of the proposed HyperLedger next generation consensus protocol
v1.0 Architecture

- **SDK**: keys 
- **membership**: No SPoF, No SPoT 
- **peer**: Endorser, Committer, Ledger, Events, Chaincode, state 
- **Consensus Service**: Order TXs in a batch according to consensus 

Next Generation Consensus Protocol

• Validation role will be split into 2 independent roles:
  – Endorsement
    • Endorsing a transaction verifying that its content obeys a given smart contract. Endorsers “sign” the contract
  – Consensus
    • Consenting the inclusion of a verified transaction in the ledger. Consensus controls what goes in the ledger making sure that the ledger is consistent

• Introduction of Endorsement Policies and Channels
Nodes and roles

• **Peer**: Commits transactions, maintains ledger and state

• **Endorsing peer**: Specialised role of peer that receives a transaction proposal for endorsement, responds granting or denying endorsement

• **Consensus-Service**: Approves the inclusion of transaction blocks into the ledger and communicates with peer and endorsing peer nodes
An endorsement policy describes the conditions by which a transaction can be endorsed. A transaction can only be considered valid if it has been endorsed according to the policy.

- Peers maintain a set of endorsement policies
- An endorsement policy is specified on deployment of chaincode
Endorsement Policy Examples

• Suppose the chaincode specifies the endorser set

  \( E = \{\text{Alice, Bob, Charlie, Dave, Eve, Frank, George}\} \).

• Example policies

  – A valid signature from all members of \( E \).
  – A valid signature from any single member of \( E \).
  – Valid signatures from a subset of \( E \) (using AND / OR)
  – Valid signatures from a subset of \( E \) based on a weighting value
Channels

Peers broadcast and receive messages from the consensus-service via channels. Channels enable partitioning of confidentiality.

- Enables chaincode confidentiality
- Messages can be partitioned into separate channels
- Nodes can connect to one or more channels
- Channels do not need to be connected to by all nodes
- Peers are permissioned to connect to a channel via an access control policy
- Transactions broadcast to a channel are ordered by the consensus service.
- All peers receive transactions in exactly the same order for a channel.
- Transactions are delivered in cryptographically linked blocks.
- Each peer validates the delivered blocks and commits them to the ledger.