

# FTEC 5520 – Week 9

---

## Agenda – Lab Session 2 – Week 9

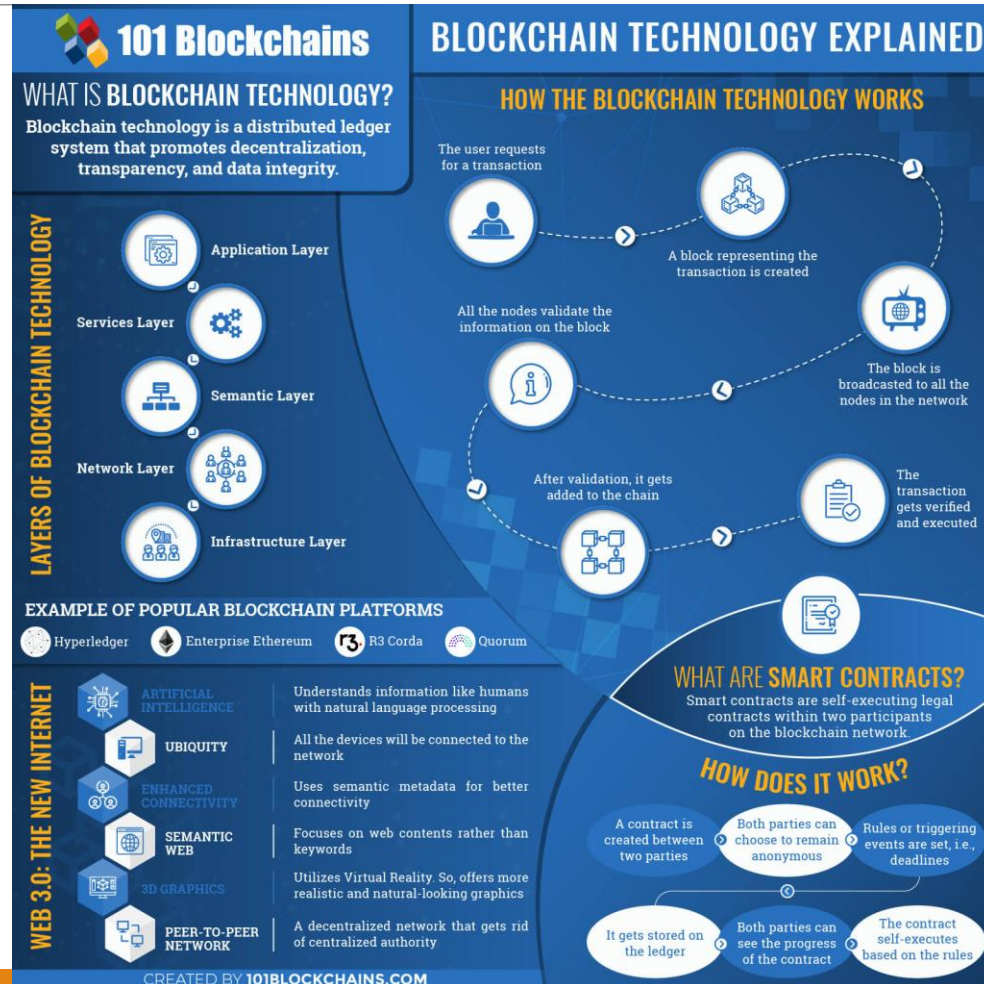
1. Another view of blockchain technology

2. Hyperledger

3. Setup and implementation of Blockchain - Hyperledger

4. Brief input about another DLT solution

# Another view of blockchain technology



# Another view of blockchain technology

---

## Layers of Blockchain technology

- Application Layer (dApps, dApp browser, User interface and application hosting)
- Services Layer (Oracles, Multi-signatures, Smart contracts, Digital Assets, Wallets, Distributed file storages, Digital identities, etc)
  - Oracles – act as an agent for collecting information from outside the network
  - Multi-signatures – Sign any transaction based on N out of M signatures for transaction
  - Smart contracts - self-executing legal contracts within two participants on the blockchain technology network
  - Digital Assets – digital version of assets that would be used in transaction such as cryptocurrencies, shares, gold, or even other kinds of document
  - Wallets – used for storing digital assets
  - Distributed file storage - distributed file storages are actually a server location where all the data will be stored
  - Digital Identity - identities of the users on the network.
- Semantic Layers (Consensus algorithm, virtual machines)
- Network Layer (Trusted Execution Environment, RLPx, Block delivery network)
- Infrastructure Layer (virtual resources such as servers, network, storage, OS, hardware, system, and server.)

# Blockchain technology

---



**Coins:** a cryptocurrency to make payments



**Tokens:**

- Cryptographic tokens are accounting units that can be used to represent the digital balance of a certain asset
- Can be created with unique data and are called nonfungible tokens (NFTs) to represent something completely unique.
- Can associate with digital or physical value of stocks, options, digital obligations, fiat currencies, ownership rights or rights for a service



**Ledgers**



**Smart contracts and DApps**

# Examples of Enterprise Blockchains

---

Ethereum is one of the popular blockchain platforms on the market suited just for enterprises.

However, as its public, it might not be suitable for blockchain technology in banking.

- Governmental support as you implements new projects based on Ethereum.
- An open platform that you can use without any issues.
- Fast upgradations to introduce newer additions and fix bugs better than others.
- Offer standards to help other companies build their very own network.

# Examples of Enterprise Blockchains

---

Hyperledger is one of the blockchain platforms that you can use in almost any kind of sector. It provides the following features

- Modular architecture that lets you plug into any kind of application and use it.
- Permissioned network that you can use to add privacy in your network.
- High scalability ensures that you enjoy the best performance for all time.
- Safety protocols that will safeguard your information.
- Data availability based on the need to know the concept.



**HYPERLEDGER**

# Enterprise World

---



# Blockchain Evolution

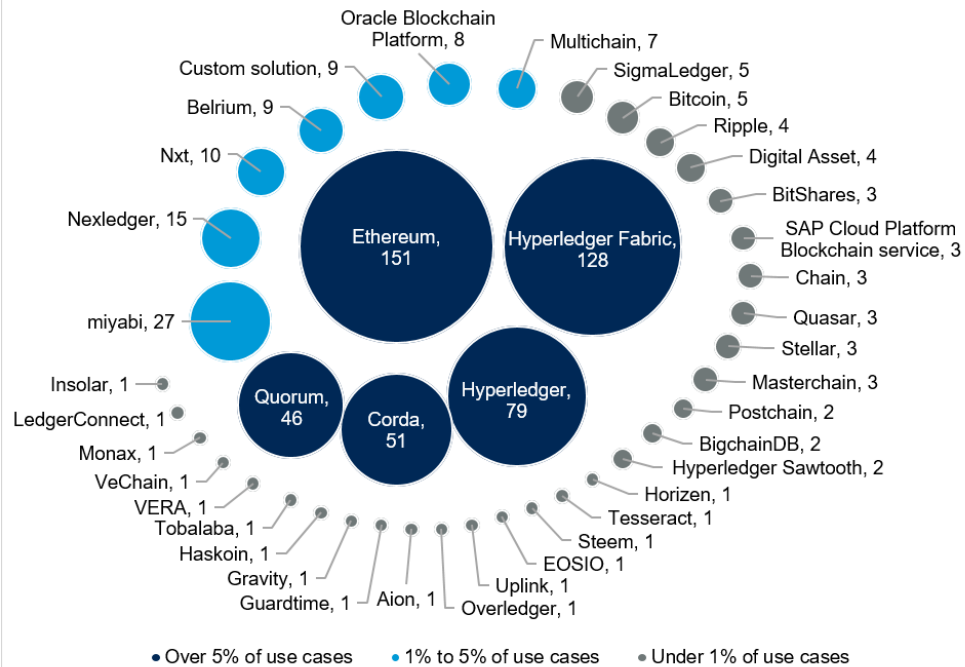
---



	Blockchain 1.0	Blockchain 2.0	Blockchain 3.0
CryptoCurrency Token	Yes	Yes (ether, user created)	None
Mining	Yes	Yes	None
Smart Contract	None	Generic	Generic
Peers	B2C (Anonymous)	B2C or B2B (private)	B2B (public or confidential)

# Primary Blockchain Technologies Utilized in POCs (2019)

## Primary Blockchain Technologies Utilized in POCs



n = 598; excludes use cases with a phase of "Other" and those where the technology could not be identified  
Source: Gartner (May 2019)  
ID: 387725

# Ethereum vs Hyperledger

	Ethereum	Hyperledger
Network	Public/Private	Consortium
Cryptocurrency	Ether	No
Consensus	Proof of Work (Public) Proof of Authority (Private)	Practical Byzantine Fault Tolerance
Smart contract	Solidity	Chaincode
Language	Go, Python	Go, Java
Maintenance	Ethereum Developer Community	Linux Foundation

# What is Hyperledger

---

**Hyperledger** is an **open source project** that came out of the Linux Foundation (LF) group and was created in order to help advance cross-industry blockchain technologies.

In **February 2016**, the Linux Foundation formally announced Hyperledger, an open source, open governance effort to advance cross-industry blockchain. It serves as a greenhouse for multiple blockchain technologies, including frameworks that implement shared, replicated ledgers, and tools for developing and operating instances of them.

One of the most advanced Hyperledger projects is called **Hyperledger Fabric**. This provides an implementation of the shared ledger and smart contract execution framework, and is built around the principles of security (to reflect the needs of regulated businesses) and modularity (to allow for innovation).

# Hyperledger Fabric

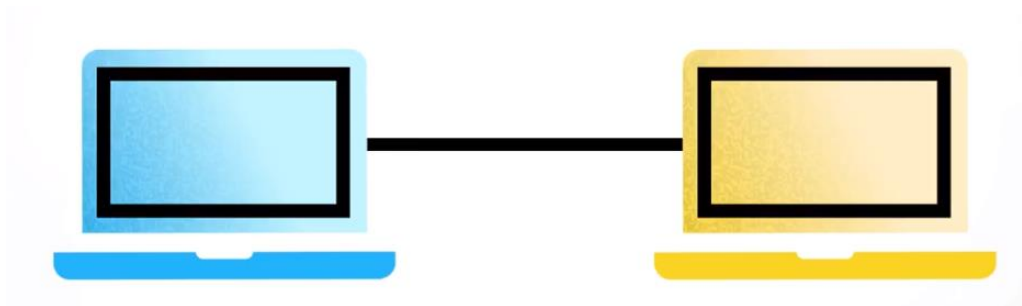
Hyperledger Fabric is a private and permissioned blockchain. Members of a network need to enrol through a trusted membership service provider (MSP).

Hyperledger Fabric do **not have build-in cryptocurrency** but can be developed with **chaincode**

All the participants of a Fabric network have known identities.

Include membership identity services

Assets to be transferred (as key-value pairs)



# Hyperledger frameworks Projects

## Consists of five blockchain frameworks



### Hyperledger Iroha

- For mobile development projects



### Hyperledger Sawtooth

- Novel consensus algorithm that Intel came up with Proof of Elapsed Time (PoET)



### Hyperledger Burrow

- Contributed by Monax and Intel
- Client-built to the specification of Ethereum Virtual Machine (EVM)



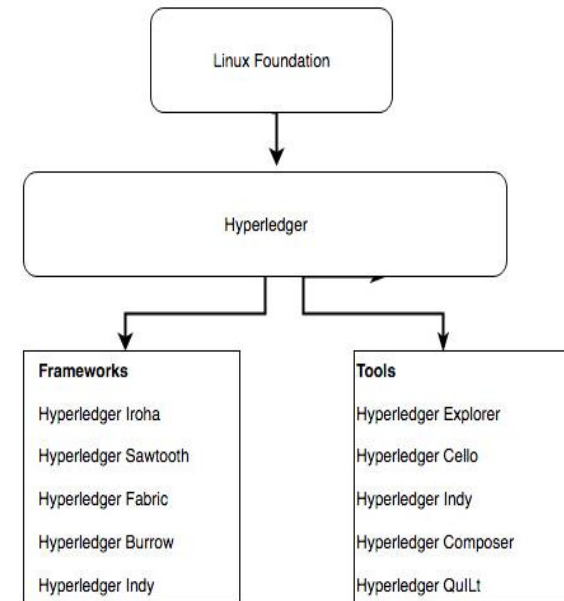
### Hyperledger Fabric

- Contributed by IBM
- Foundation for developing applications or solutions
- Leverages containers to host smart contracts called chaincode



### Hyperledger Indy

- Contributed by Sovrin foundation
- Support independent identity on distributed ledgers

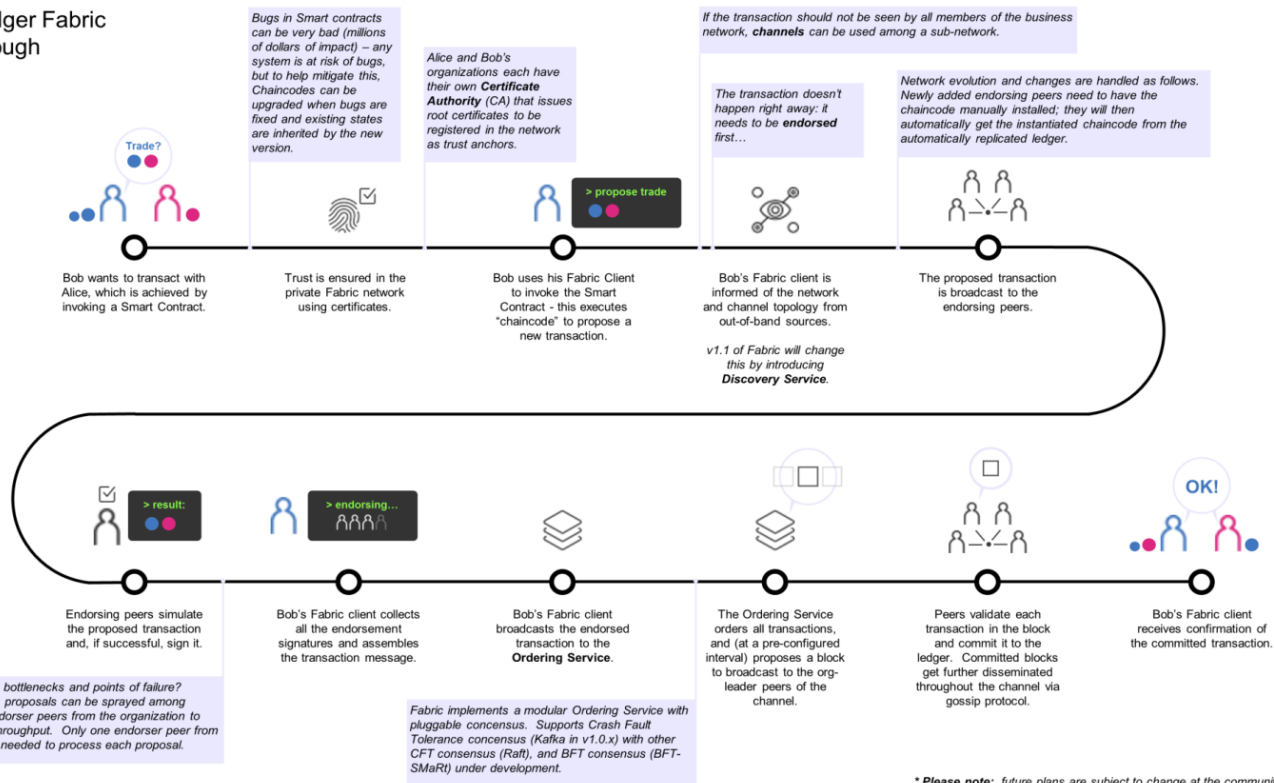


## Additional Hyperledger components:

- Indy – identity management tools
- Quilt – interoperability tools
- Cello – provisioning tools
- Composer – development tools
- Explorer – inspection tools

# Hyperledger Fabric Walkthrough

## Hyperledger Fabric Walkthrough



\* Please note: future plans are subject to change at the community's discretion

# Benefit of using Hyperledger Fabric

---



- **Increased privacy and enhanced security**

- Hyperledger fabric uses hardware security module (HSM) and thus ability to safeguard data and provide unique decryption keys is easier. This provides a strong process of authentication.



- **Provide multiple plug-ins for customization**

- Hyperledger fabric provides enterprise blockchain developers with the ability to use a modular architecture that supports multiple plug-ins and components.



- **No digital identity theft with permissioned membership**

- With hyperledger fabric, enterprise blockchain developers can provide you with an infrastructure where all the stored data becomes a need to know basis.



- **Hyperledger composer for enterprise blockchain development**

- Hyperledger composer has the ability to combine different tools together and provide you with enhanced smart contracts that increase the rate of operability for any and all of your business functions.



- **Enhanced collaboration between developers**

- Hyperledger composer has a pool of existing guidelines and experts with high-skills and experience in enterprise hyperledger blockchain development.



## Issues in Hyperledger Fabric

---

- 1) hashes on chain cannot be validated by any third party, so they can be used by adversaries to trick honest participants
- 2) private data use gossip to transact data, which would require all participants be connected with any other participant part of a chain. if there are 20 participants in a channel, each participant must open up their firewalls to all other 19 participants of a single channel
- 3) hashes put on chain don't have salt added to it, which is vulnerable to dictionary attack
- 4) when using k8s and behind load-balancers or proxies, users do not even get a chance to use a shared port (in the aforementioned example, each participant can't even open firewalls to 19 other participants without extensive hacking, and I assumed all participants need to deployed these hacked code to make it work

# Hyperledger Fabric 2.0 and Privacy

---

Hyperledger Fabric 2.0 supports the ability to perform transactions with private data.

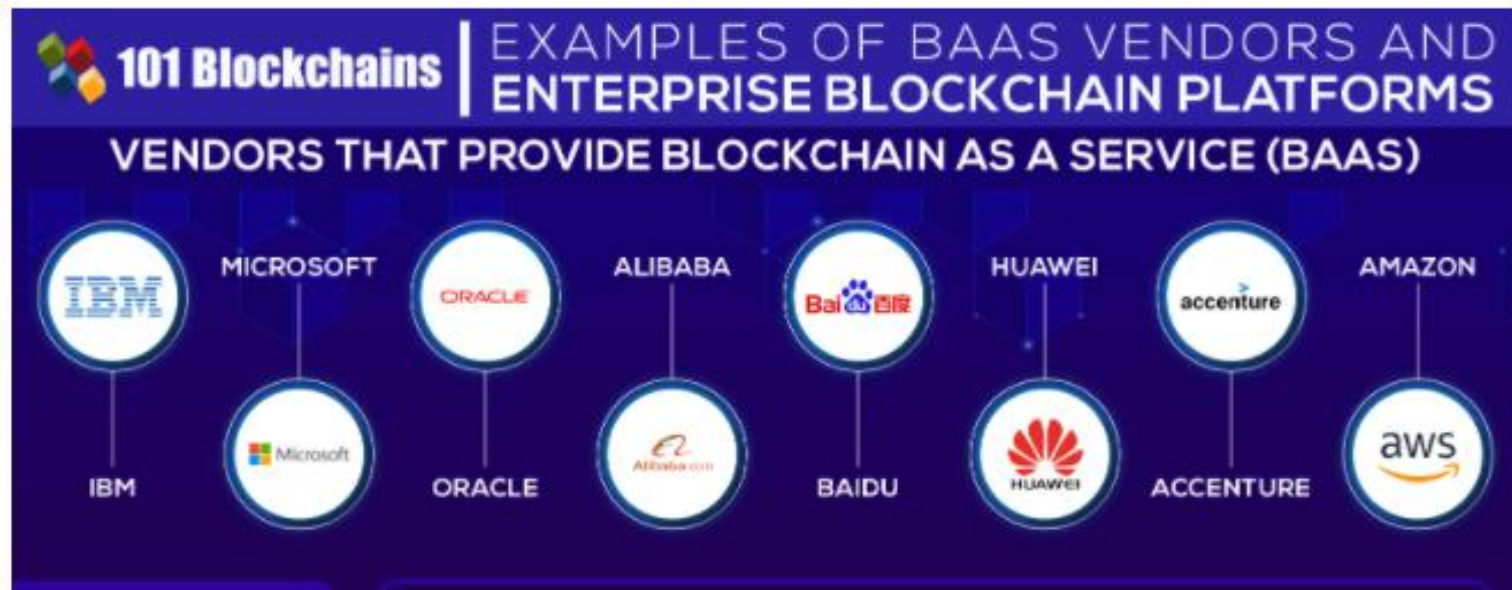
Within Hyperledger there is work being performed on Zero Knowledge Proofs (ZKP).

Zero-knowledge Proof is an encryption scheme proposed by MIT researchers Silvio Micali, Shafi Goldwasser, and Charles Rackoff in the 1980s.

In this method, one party (Prover) can prove that a specific statement is true to the other party (Verifier) without disclosing any additional information.

# Choices of Blockchain Implementation

**BAAS or Blockchain as a Service** is a special kind of enterprise-grade blockchain solution that other customers can utilize to host, build, and own their developed decentralized apps, functions and other kinds of smart contracts.




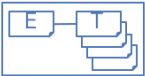






# BaaS Providers

BaaS Providers	Solutions	Supported DLT
Oracle	Enterprise blockchain solution	Hyperledger Fabric
IBM	Fully functional, integrated, enterprise-ready blockchain solution	Hyperledger Fabric Architecture
Amazon	AWS Blockchain Templates on ECS or ECS2	Ethereum or Hyperledger Fabric
Azure	Azure Blockchain Workbench	DLT built on top of SQL, Hyperledger Fabric, Ethereum PoA
Alibaba	Alibaba Cloud BaaS API	Hyperledger Fabric, Quorum, Ant Blockchain
Baidu	Blockchain platform	Xuper Chain
Huawei	Pre-built on Hyperledger Fabric 1.1 with K8s	Hyperledger Fabric
Accenture	No	

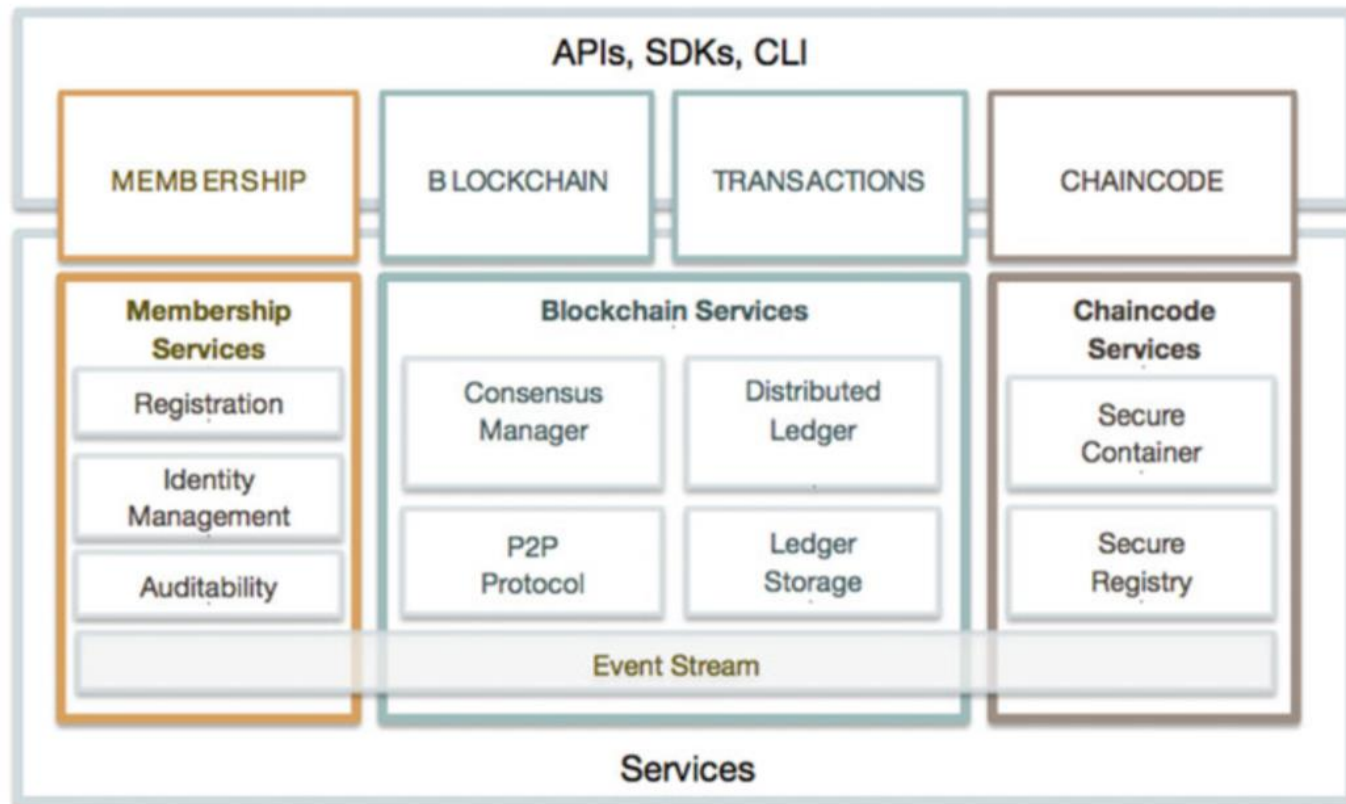
# Components in Blockchain network

---

Ledger		A ledger is a channel's chain and current state data which is maintained by each peer on the channel.
Smart Contract		Software running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets.
Peer Network		A broader term overarching the entire transactional flow, which serves to generate an agreement on the order and to confirm the correctness of the set of transactions constituting a block.
Membership		Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network.
Events		Creates notifications of significant operations on the blockchain (e.g. a new block), as well as notifications related to smart contracts.
Systems Management		Provides the ability to create, change and monitor blockchain components
Wallet		Securely manages a user's security credentials
Systems Integration		Responsible for integrating Blockchain bi-directionally with external systems. Not part of blockchain, but used with it.

# Hyperledger Fabric Architecture

---



# Hyperledger Fabric Design

A Hyperledger Fabric network consists of the following components:

- **Assets**: Assets are key-value pairs that represent a value. A value can be anything such as a document, stock, or cryptocurrency token. Each asset holds a state and ownership.
- **Shared ledger**: A shared ledger holds its own copy of the ledger with the state of the asset. This ledger is called the world state. The shared ledger also holds a copy of the blockchain, which stores the ownership of the asset by recording the transaction's history.
- **Smart contracts** (chaincode): Hyperledger Fabric calls smart contracts chaincodes that can be programmed in Go (GoLang) or JavaScript (Node.js). Chaincode can interact with the shared ledger, assets, and transactions.
- **Membership services provider** (MSP) : The MSP is the certificate authority that manages the digital certificate
- **Peer nodes**: The Hyperledger Fabric network is built on peer nodes that are owned and contributed by members of the network.
- **Channel**: Channels can be created by a collection of peer nodes.
- **Organizations**: Each peer node contributes resources, and together they form the collective network
- **Ordering service**: This service packages transactions into blocks.

Excerpt From: Elad Elrom. "A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects". Apple Books.

# Hyperledger Fabric Architecture

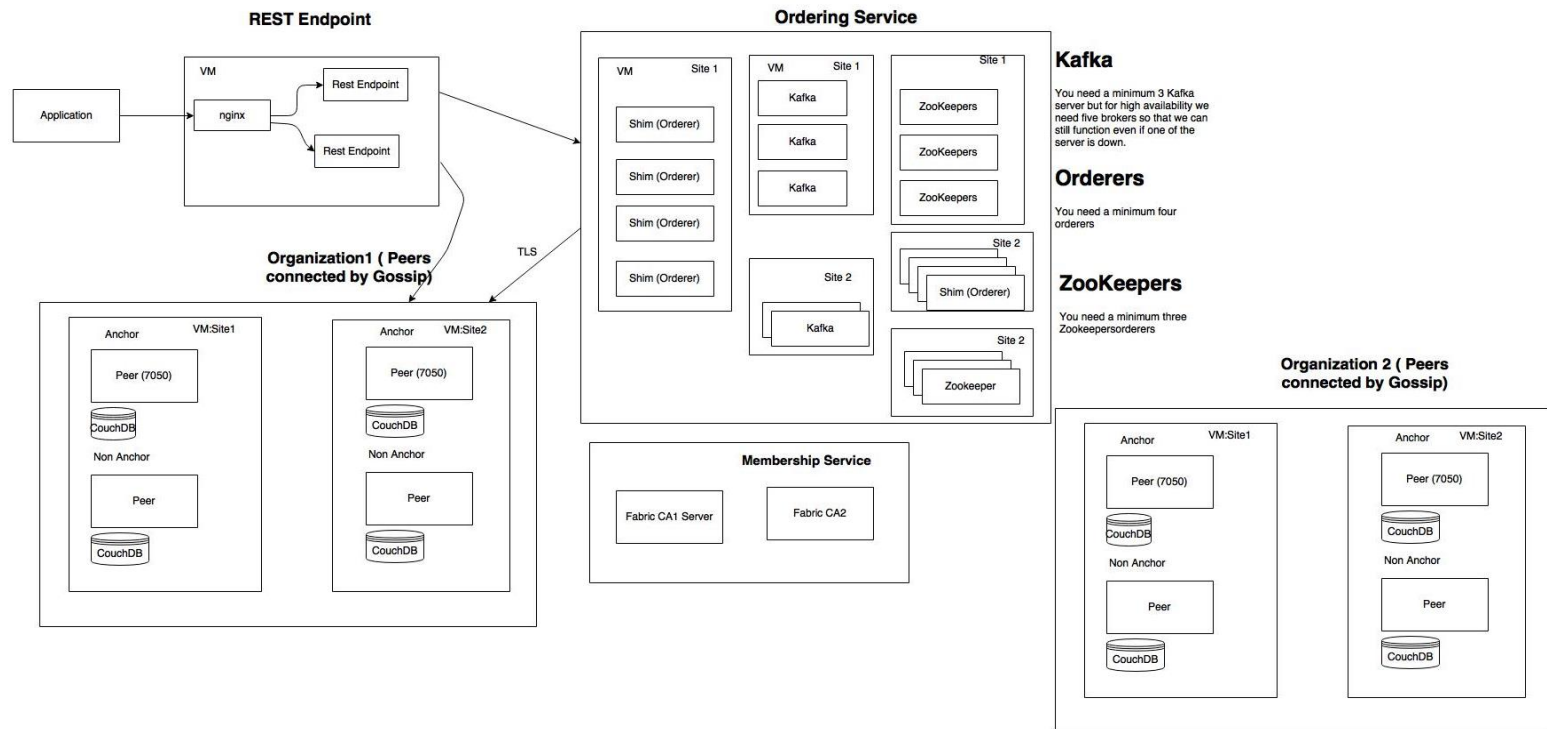
---

The distributed ledger protocol of the fabric is run by peers. The fabric distinguishes between two kinds of peers:

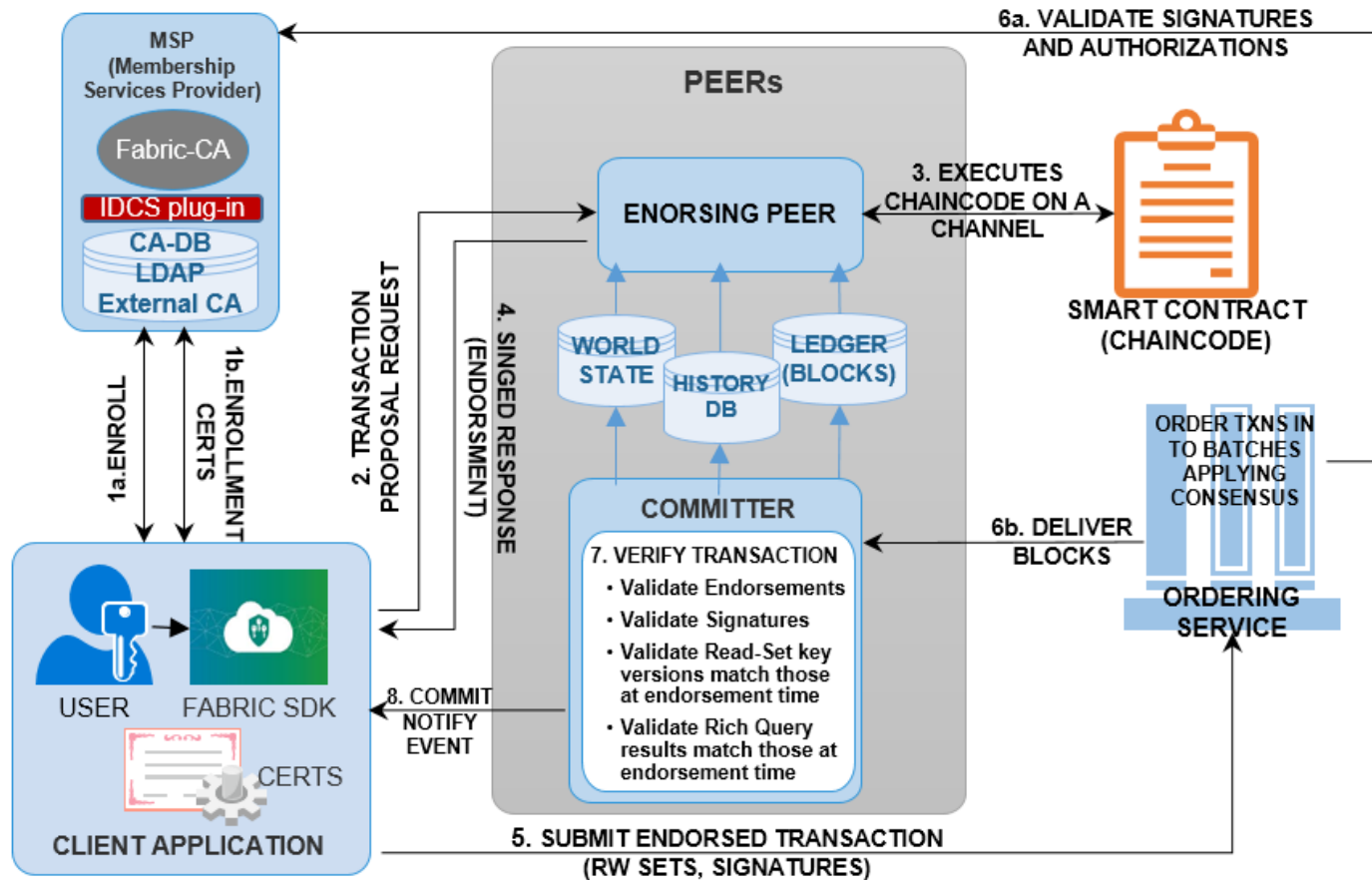
- **Validating peer**: is a node on the network responsible for running consensus, validating transactions, and maintaining the ledger.
- **Non-validating peer**: is a node that functions as a proxy to connect clients (issuing transactions) to validating peers. A non-validating peer does not execute transactions but it may verify them.



# Hyperledger Fabric Infrastructure



# Logical Flow diagram of Hyperledger



From <https://www.oracle.com/webfolder/s/assets/ebook/developing-dapps-oracle-blockchain/index.html#/page/9>

# Hyperledger Fabric Structure

---

# Hyperledger Fabric Infrastructure

## Three types of nodes within a Hyperledger Fabric system:

### Client:

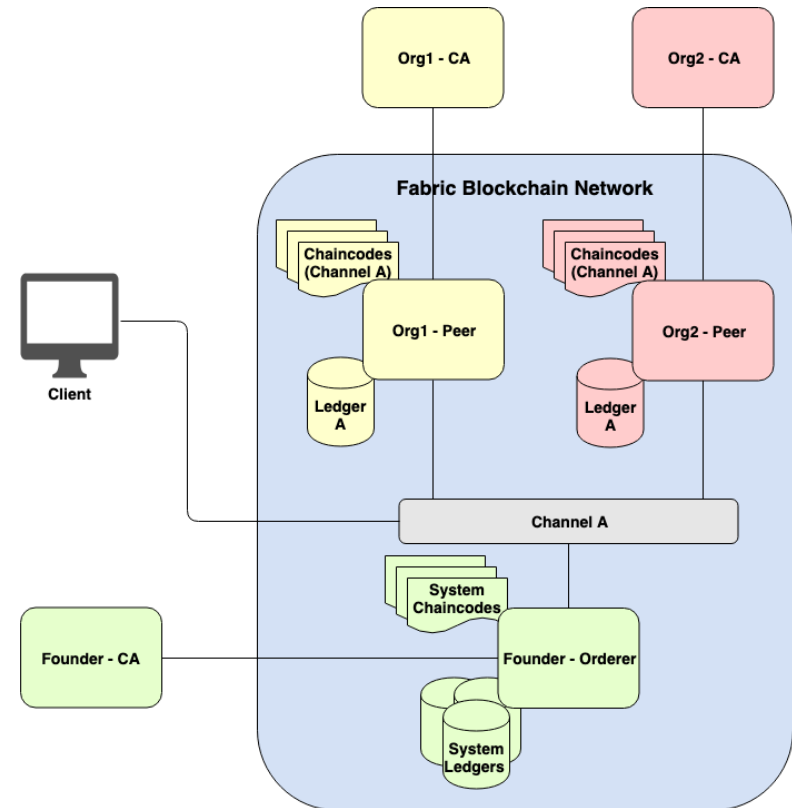
- A client acts on behalf of an end user. It connects to peers to communicate with the blockchain.

### Peer:

- A peer node receives ordered state updates in the form of transactions from the orderers, **execute and verify, commits transactions, and maintains the state of the ledger.** (Anchor peer and Endorser peer)

### Orderer:

- An orderer node **validates the transactions** based on the endorsement policy and **orders the transactions into a sequence before broadcasting** them into the network.



\*\*\* Actually, the system chaincodes and ledgers are also deployed on peers but they were not put into the figure for the sake of simplicity \*\*\*

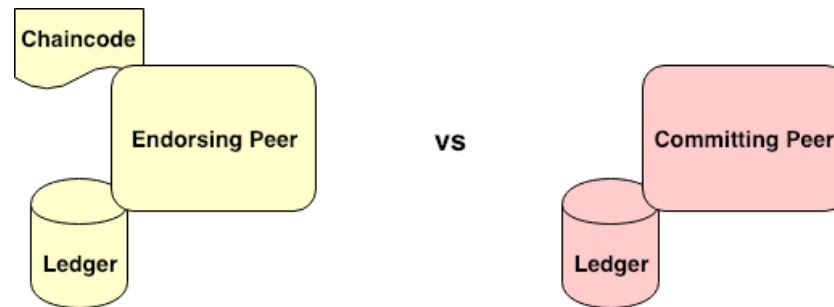
# Hyperledger Fabric Infrastructure

There is a concept of Smart Contract called Chaincode in Fabric.

Currently, three languages can program Fabric chaincode including Golang, Node.js, and Java.

To deploy a chaincode, a network admin must install the **chaincode onto target peers** and then invoke an orderer to instantiate the chaincode onto a specific channel.

Endorsement policy defines which peers need to agree on the results of a transaction before the transaction can be added onto ledgers of all peers on the channel.



# Hyperledger Fabric Infrastructure - Ledger

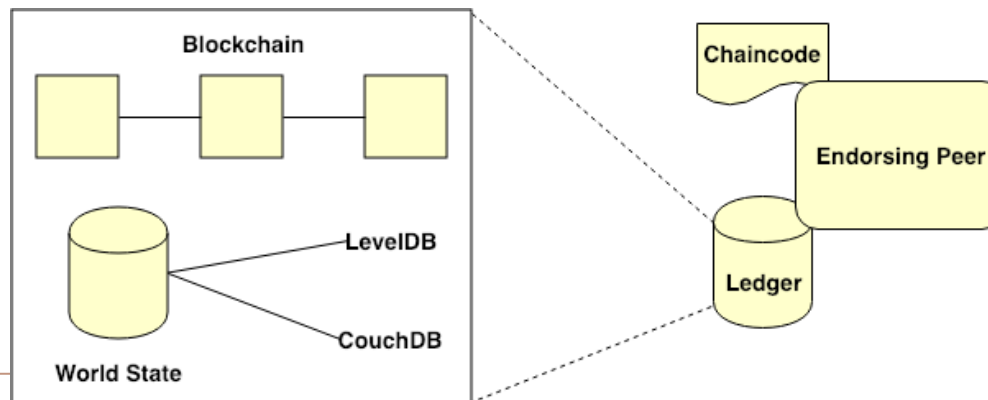
The interior components inside the Peer's ledger include Blockchain and World State.

- Blockchain holds the **history of all transactions** (Transaction Log) for every chaincode on a particular channel.
- World State maintains the current state of variables for each specific chaincode.

**Transaction Logs are written into the blockchain**

**Two types of World State database** currently supported in Fabric include

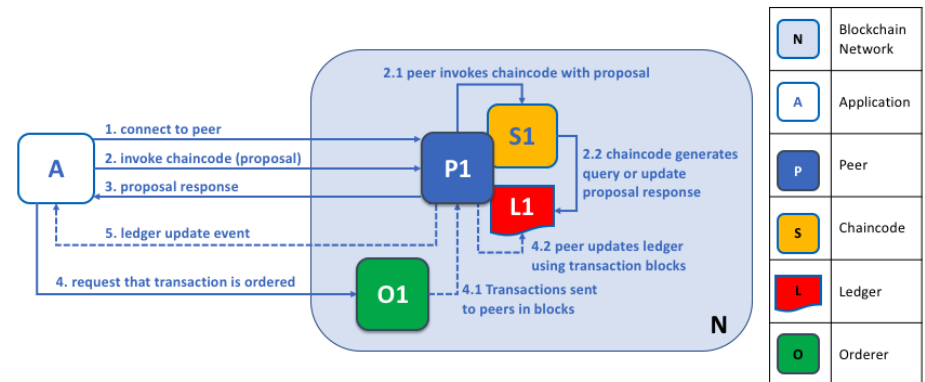
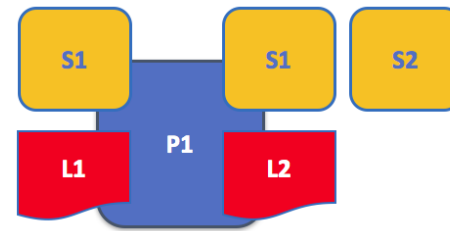
- **LevelDB and CouchDB.**
- LevelDB is a default **key-value database built** on Fabric Peer
- CouchDB is a **JSON-based database** supporting rich querying operations based on JSON objects.



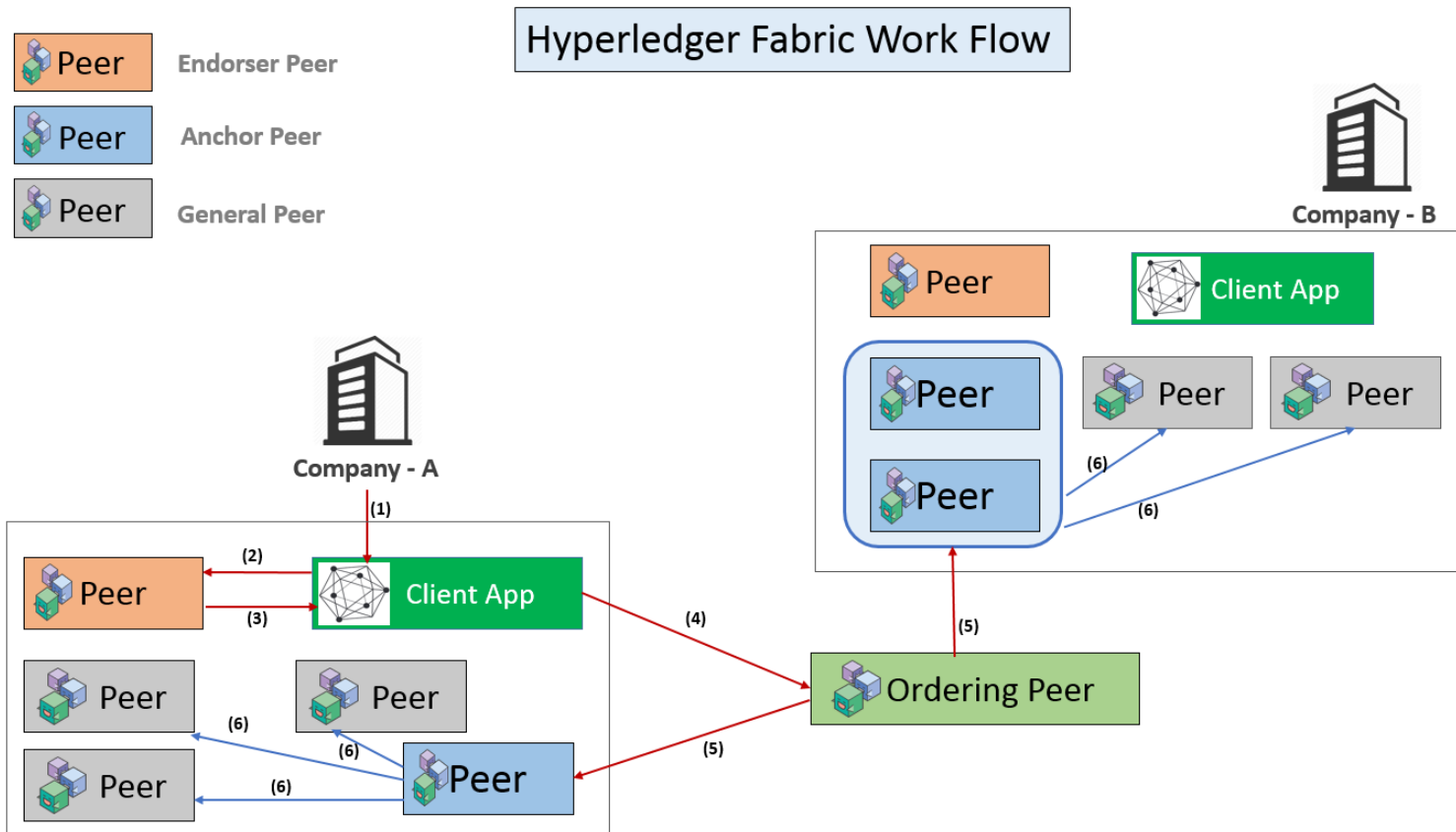
# Hyperledger Fabric Infrastructure - Ledger

A peer is able to host more than one ledger, which is helpful because it allows for a flexible system design.

The simplest configuration is for a peer to manage a single ledger, but it's absolutely appropriate for a peer to host two or more ledgers when required.

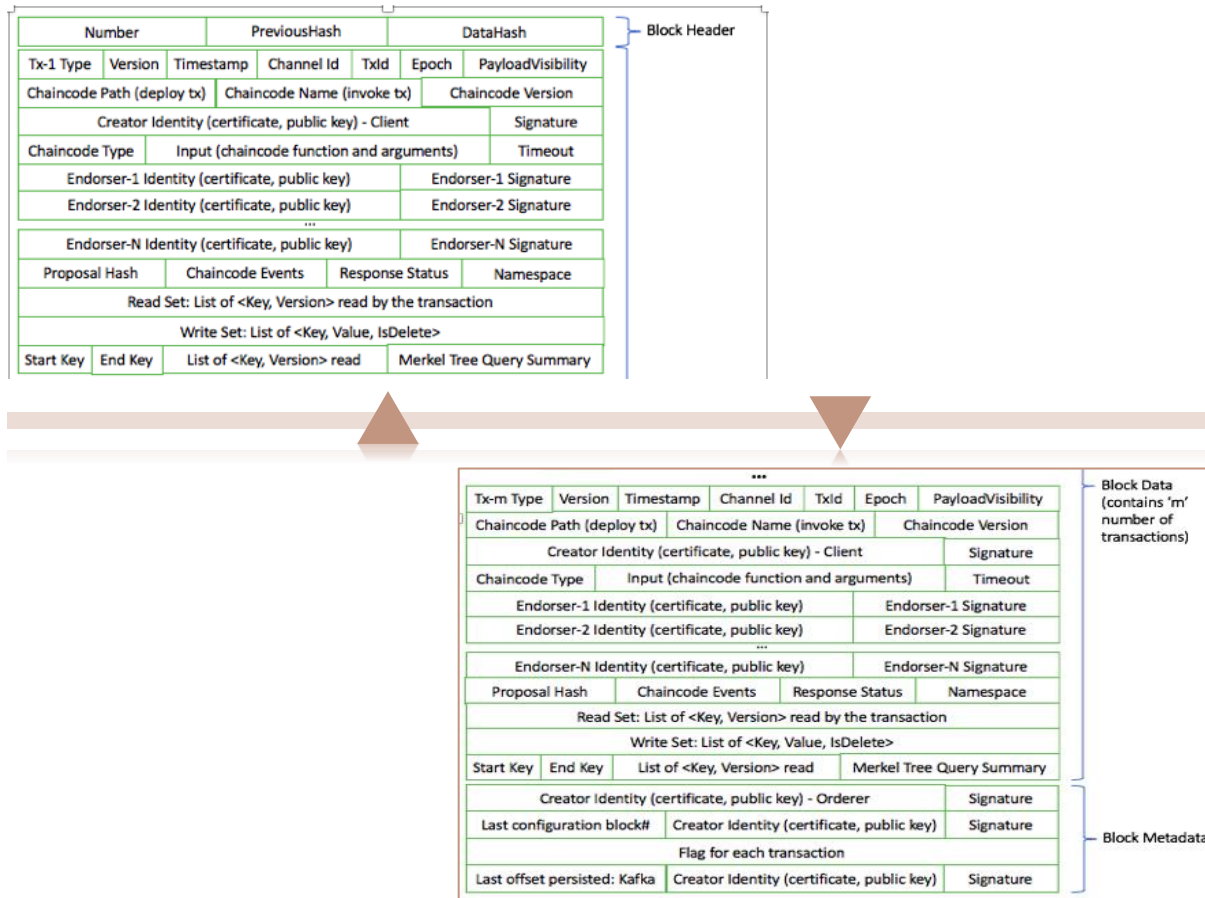


# Hyperledger Fabric Workflow



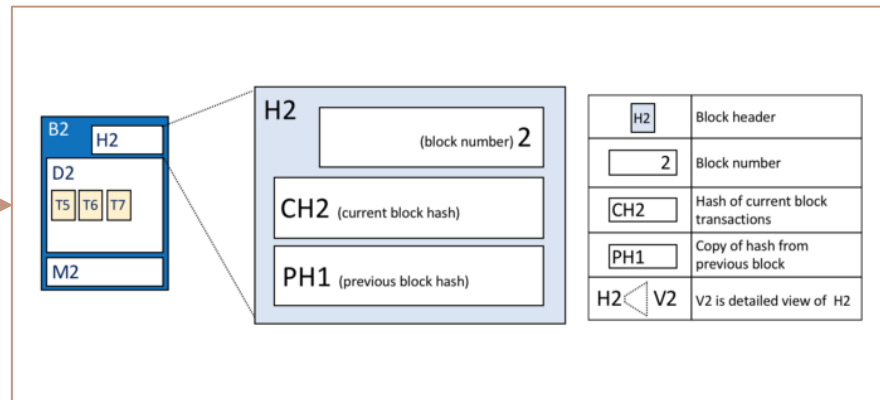


# Hyperledger Fabric Block structure

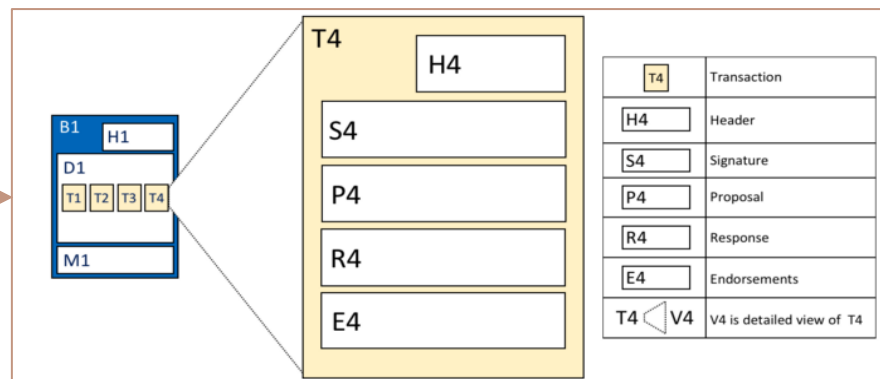


# Hyperledger Fabric Block structure

Header



Transaction (Body)



# Key Characteristics of Enterprise Blockchain

---



# Smart Contracts (Chaincode)

---

Hyperledger Fabric has taken a **generic approach to the languages** used to write smart contracts.

Assets to be processed through transaction and controlled through chaincode

Although the industry uses and understands the generic term smart contract, Hyperledger Fabric actually refers to the installable unit as chaincode. We can think of smart contracts as being equivalent to chaincode.

Hyperledger Fabric leverages container technology to host smart contracts called chaincode that comprise the application logic of the system. Chaincode can be implemented in programming languages such as **Go or Java** and is invoked through a transaction proposal.

The first supported language was **Go**, which is also the **language that Fabric** itself is written in. Node.js and Java have been added in later versions. Hyperledger Burrow™ has also been added to Hyperledger Fabric, which adds languages such as Ethereum's Solidity.

# Chaincode

---

Chaincode is a program, written in **Go, node.js, or Java** that implements a prescribed interface.

It runs in a **secured Docker container** isolated from the **endorsing peer process**.

It initializes and manages ledger state through transactions submitted by applications.

# Chaincode Sample (NGO AWS sample)

```
let Chaincode = class {  
  
    /**  
     * Initialize the state when the chaincode is either instantiated or upgraded  
     *  
     * @param {*} stub  
     */  
    async Init(stub) {  
        console.log('===== Init: Instantiated / Upgraded ngo chaincode =====');  
        return shim.success();  
    }  
}
```

```
docker exec -e "CORE_PEER_TLS_ENABLED=true" -e  
"CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \ -e  
"CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" -e  
"CORE_PEER_ADDRESS=$PEER" \ cli peer chaincode instantiate -o $ORDERER -C  
mychannel -n ngo -v v0 -c '{"Args":["init"]}' --cafile /opt/home/managedblockchain-tls-  
chain.pem --tls
```

# Chaincode Sample (NGO AWS sample)

```
/**                                     /**
 * Retrieves a specific donor          * Retrieves all donors
 */
 *
 * @param {*} stub
 * @param {*} args
 */
async queryDonor(stub, args) {
  console.log('===== START : queryDonor =====');
  console.log('##### queryDonor arguments: ' +
    JSON.stringify(args));

  // args is passed as a JSON string
  let json = JSON.parse(args);
  let key = 'donor' + json['donorUserName'];
  console.log('##### queryDonor key: ' + key);
  return queryByKey(stub, key);
}

/**                                     /**
 *
 * @param {*} stub
 * @param {*} args
 */
async queryAllDonors(stub, args) {
  console.log('===== START : queryAllDonors =====');
  console.log('##### queryAllDonors arguments: ' + JSON.stringify(args));

  let queryString = '{"selector": {"docType": "donor"}}';
  return queryByString(stub, queryString);
}
```

`docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \ -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \ cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryAllDonors"]}'`

`docker exec -e "CORE_PEER_TLS_ENABLED=true" -e "CORE_PEER_TLS_ROOTCERT_FILE=/opt/home/managedblockchain-tls-chain.pem" \ -e "CORE_PEER_ADDRESS=$PEER" -e "CORE_PEER_LOCALMSPID=$MSP" -e "CORE_PEER_MSPCONFIGPATH=$MSP_PATH" \ cli peer chaincode query -C mychannel -n ngo -c '{"Args":["queryDonor","{\donorUserName\": \"edge\"}"]}'`

# Consensus

Hyperledger Fabric uses a permissioned **voting-based consensus** which assumes that all participants in the network are partially trusted. The consensus can be divided into three phases as follow.

## □ (Steps 1–3)

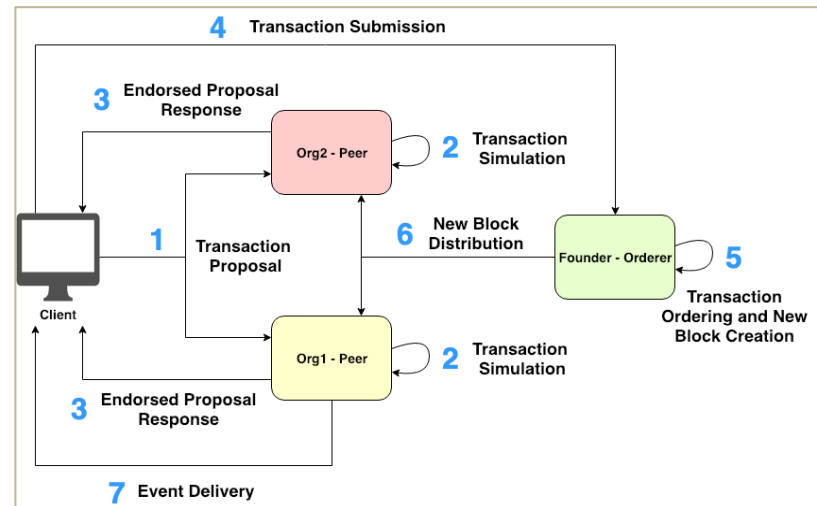
*Endorsement*

## □ (Steps 4–5)

*Ordering*

## □ (Step 6)

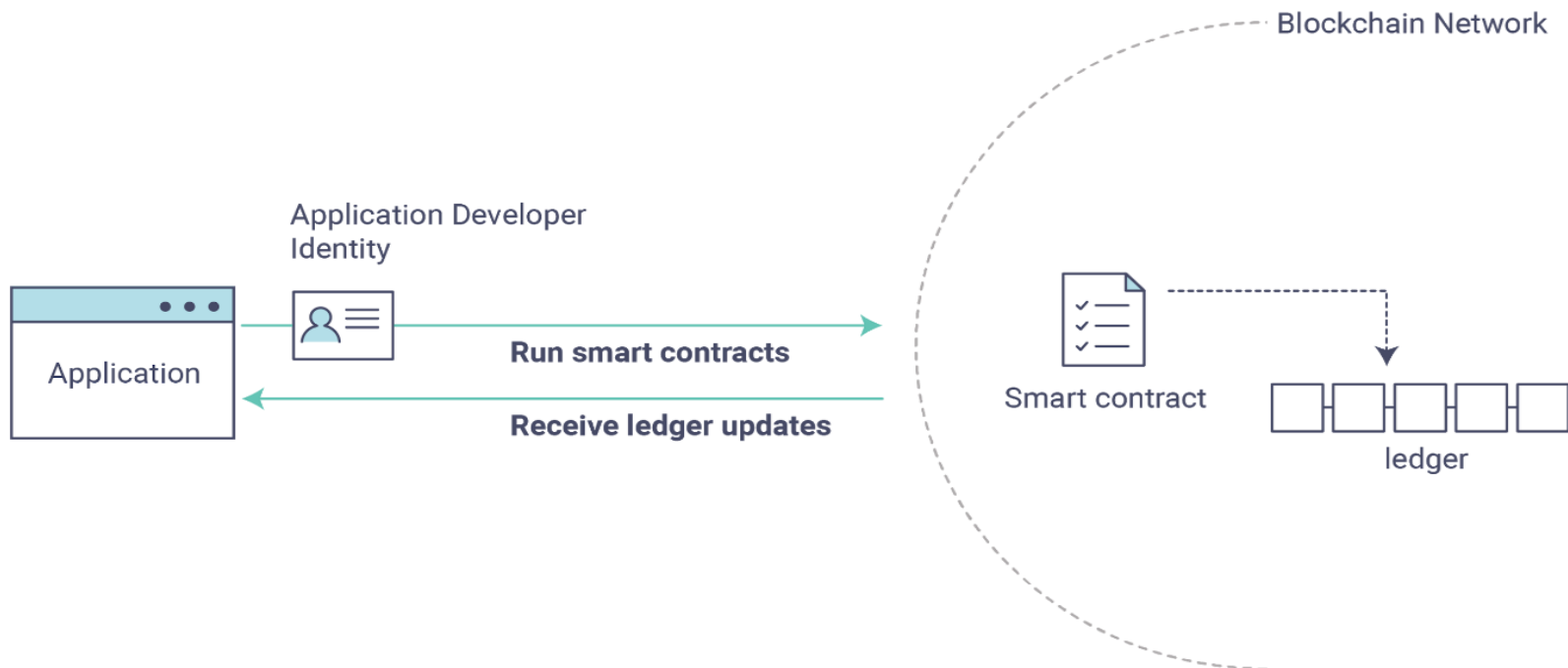
*Validation and Commitment*



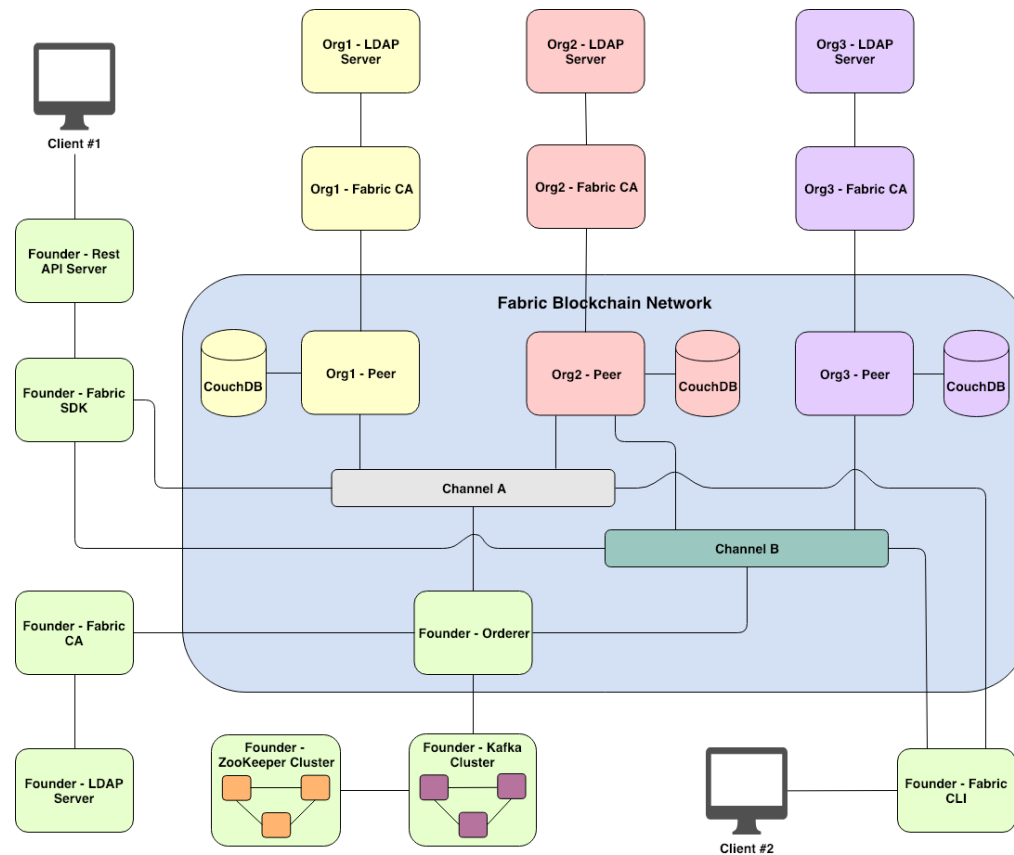


# D-Apps (Decentralized Apps) @ HyperLedger

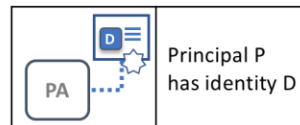
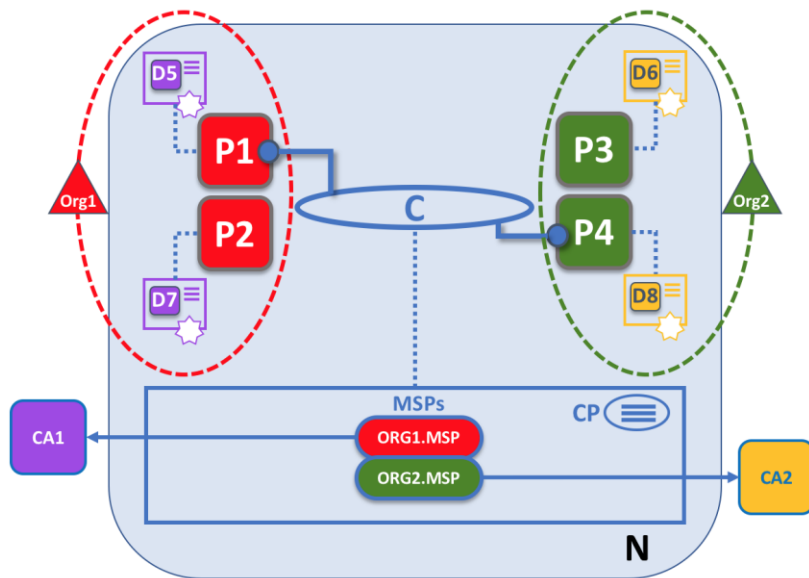
---



# An example of Hyperledger Fabric case



# Peers and Identity



<b>N</b>	Blockchain Network	<b>P</b>	Peer
<b>C</b>	Channel	<b>Org</b>	Organization
<b>I</b>	Identity	<b>PA</b>	Principal PA (e.g. P1,P4) communicates via channel C.
<b>CP</b>	Channel policy	<b>C</b>	
<b>CA</b>	Certificate Authority	<b>MSP</b>	Membership Service Provider
		Organization R owns application A1 and peers P1, P2.	
	Channel C subject to policy CP.		Channel policy CP contains MSPs: MSP1 and MSP2.
	MSP1 selects the Certificate Authority CA1 to provide certificates for it.		

# DLT and Other types of Blockchain

---

# Distributed Ledger - Blockchain

## Distributed Blockchain

## Peer A

The diagram illustrates a sequence of three blocks in a blockchain. Each block is represented by a light green rounded rectangle with a darker green border. The first block is labeled 'Block: # 1' and contains a 'Nonce: 11316' and a large empty 'Data:' field. Below it is a 'Prev:' field with a long hexadecimal string and a 'Hash:' field with a shorter hexadecimal string. A blue 'Mine' button is at the bottom. The second block is labeled 'Block: # 2' and contains a 'Nonce: 35230' and a large empty 'Data:' field. Its 'Prev:' field contains the hash of the first block, and its 'Hash:' field contains a new hexadecimal string. A blue 'Mine' button is at the bottom. The third block is labeled 'Block: # 3' and contains a 'Nonce: 12937' and a large empty 'Data:' field. Its 'Prev:' field contains the hash of the second block, and its 'Hash:' field contains a new hexadecimal string. A blue 'Mine' button is at the bottom.

## Peer B

Block:

# 1

Nonce:

11316

Data:

Proof:

Block:

# 2

Nonce:

35230

Data:

Proof:

Block:

# 3

Nonce:

12937

Data:

Proof:

<https://anders.com/blockchain/distributed.html>

# Distributed Ledger Technology

---

## Ledger

the technology uses an append only ledger to provide full transactional history

## Centralized Ledger

Processed through centralized database

## Distributed Ledger

A distributed ledger is a database of replicated, shared, and synchronized digital data that is geographically spread across multiple sites in a network.

Rather than having a central administrator like a traditional database, (think banks, governments & accountants), distributed ledgers have a system of synchronized databases that provide an auditable history of information and are visible to anyone within the network.

## Blockchain

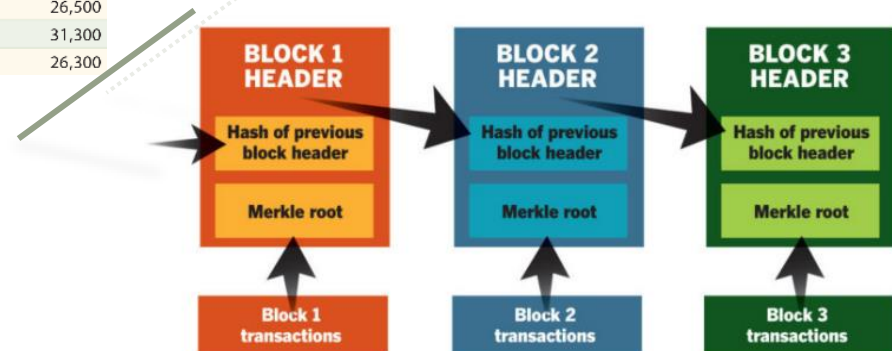
Distributed ledger technology and blockchain share the same conceptual origin and purpose  
Distributed ledgers rely on similar principles of consensus to a blockchain.

# Distributed Ledger Technology

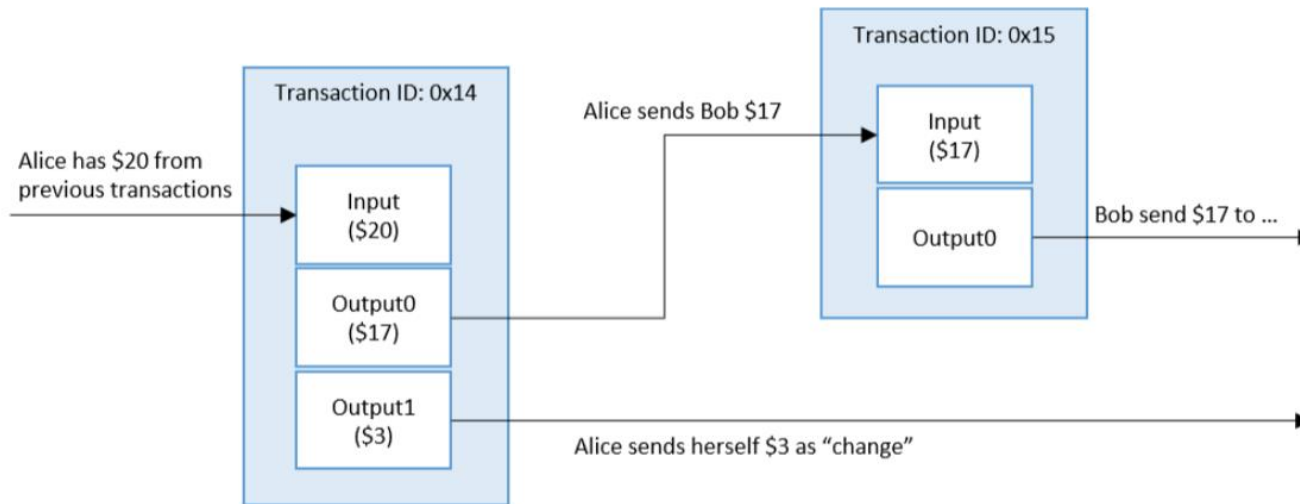
## TRADITIONAL LEDGERS

Cash				
Date	Description	Debit	Credit	Balance
Jan. 1, 20X3	Balance forward			\$ -
Jan. 1, 20X3	Journal page 1	\$ 25,000		25,000
Jan. 4, 20X3	Journal page 1		\$ 2,000	23,000
Jan. 8, 20X3	Journal page 1	4,000		27,000
Jan. 18, 20X3	Journal page 2		500	26,500
Jan. 25, 20X3	Journal page 2	4,800		31,300
Jan. 28, 20X3	Journal page 2		5,000	26,300

## CHAIN OF BLOCKS (BLOCKCHAIN)



# Example of Cryptocurrency Transaction Ledger



Example Cryptocurrency Transaction Ledger from NISTIR 8202



# Transaction Example

## Bitcoin Transaction Example

txid 90b18aa54288ec610d83ff1abe90f10d8ca87fb6411a72b2e56a169fdc9b0219

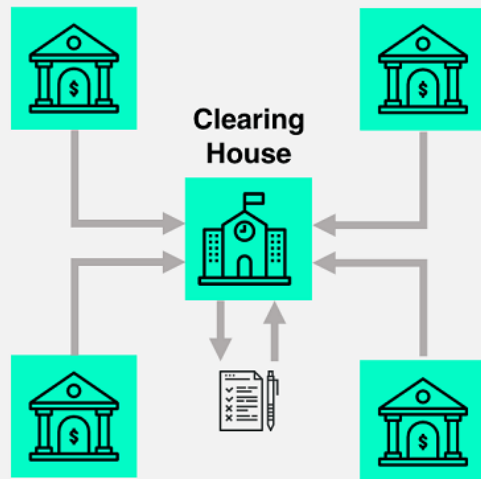
**TxID**

```
{
  "hash": "90b18aa54288ec610d83ff1abe90f10d8ca87fb6411a72b2e56a169fdc9b0219",
  "ver": 1,
  "vin_sz": 1,
  "vout_sz": 2,
  "lock_time": 0,
  "size": 226,
  "in": [
    {
      "prev_out": {
        "hash": "18798f8795ded46c3086f48d5bdabe10e1755524b43912320b81ef547b2f939a",
        "n": 0
      },
      "scriptSig": "3045022100c1efcad5cdcc0dcf7c2a79d9e1566523af9c7229c78ef71ee8b6300ab...[snip]"
    }
  ],
  "out": [
    {
      "value": "5.93100000",
      "scriptPubKey": "OP_DUP OP_HASH160 4b358739fc7984b8101278988beba0cc00867adc OP_EQUALVERIFY OP_CHECKSIG"
    },
    {
      "value": "1678.06900000",
      "scriptPubKey": "OP_DUP OP_HASH160 55368b388ccfe22a3f837c9eee93d053460db339 OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
}
```

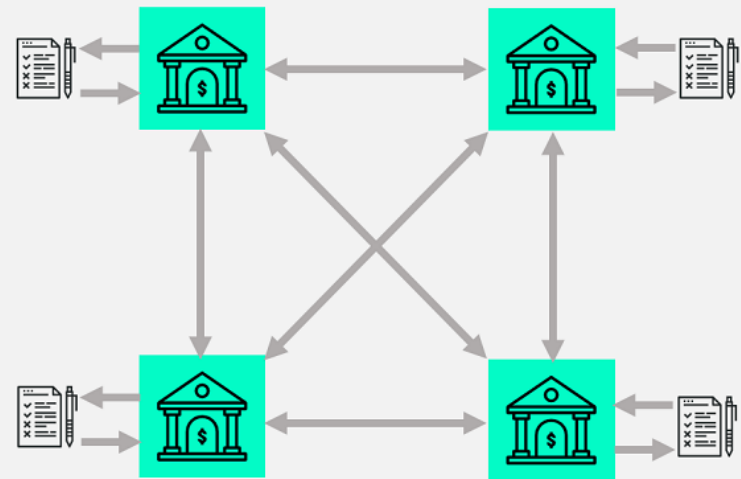
tx format version - currently at version 1  
in-counter - number of input amounts  
out-counter - number of output amounts  
tx lock\_time - should be 0 or in the past for the tx to be valid and included in a block  
size - of the transaction in bytes

image by Venzen <venzen@mail.bihthai.net> 2014 CC SA  
conditions of reuse: <http://sofala.bihthai.net/works/txinout.htm>

# Distributed Ledger Technology

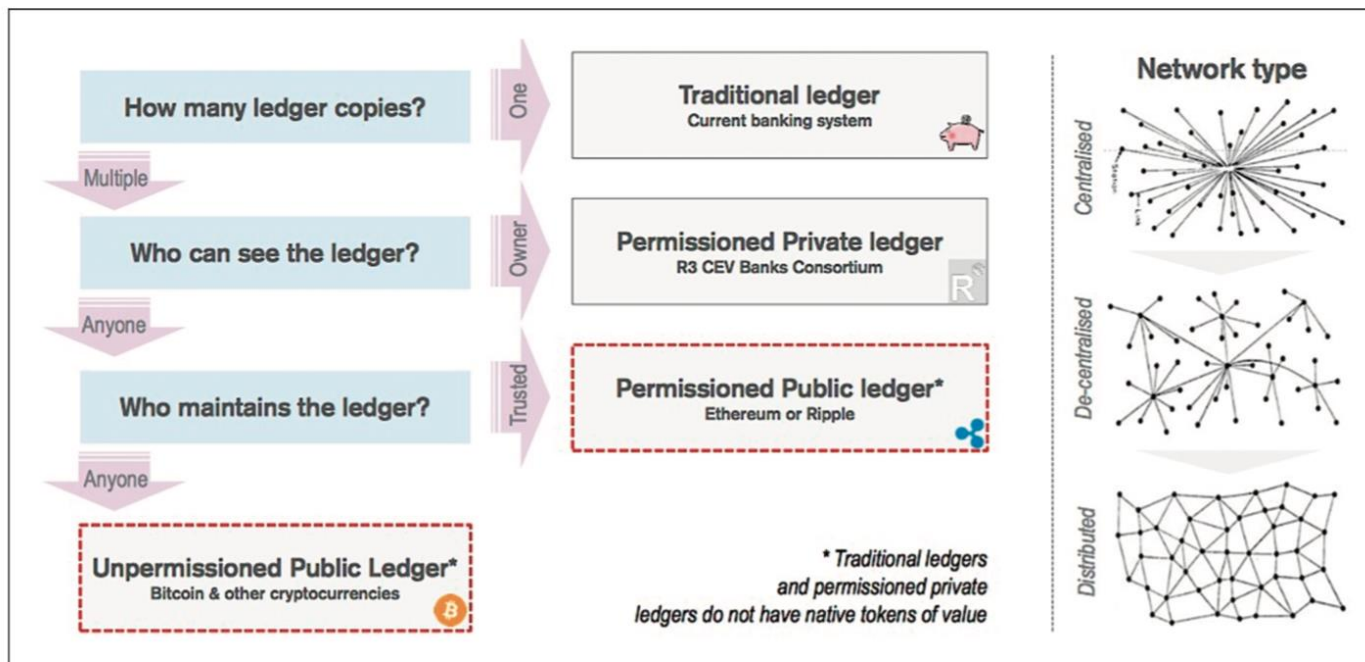


**Centralized Ledger**



**Distributed Ledger**

# Distributed Ledger Technology



**Figure 2.** Understanding blockchain types (Credit Suisse, 2016).

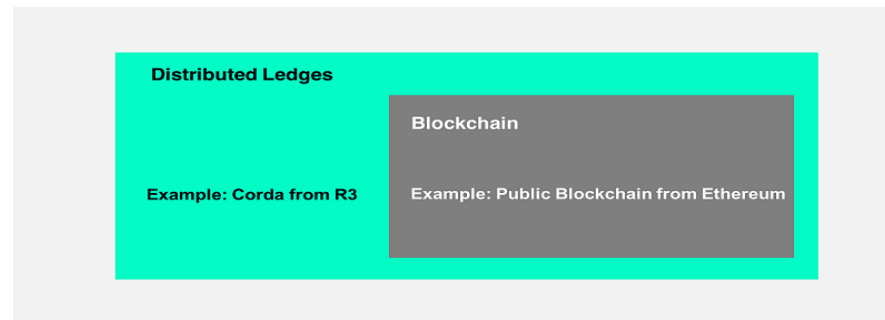
What is the purpose of Blockchain?

# Distributed Ledger Technology and Blockchain

A distributed ledger is a record of consensus with a **cryptographic audit trail** which is maintained and validated by several separate nodes

Distributed ledgers can be either decentralized, granting equal rights within the protocol to all participants or centralized, designating certain users particular rights. Distributed ledgers do not require such a chain

Blockchain is a **sequence of blocks**. A blockchain data structure has a shared, replicated ledger comprised of digitally recorded and unchangeable data in packages called blocks.



# Distributed Ledger Technology

---

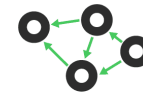
## More common types of DLTs such as

- Blockchain,
- DAG
- T-DAG
- Byteball (<https://byteball.org>)
- Nano (<https://raiblocks.net>)
- HashGraph (<https://www.hederahashgraph.com>)
- Hycon ([hycon.io](https://hycon.io))
- IoT Chain ([iotchain.io](https://iotchain.io))

## Types Of Distributed Ledger Technologies



Blockchain



Directed Acyclic  
Graph



Hashgraph

# Directed Acyclic Graph

DAG has no blocks, chains, miners, or transaction fees.

DAG is a **graph that travels in one direction** without cycles connecting the other edges.

Transactions are **not added into blocks**. It can be added as **random** or any defined algorithm.

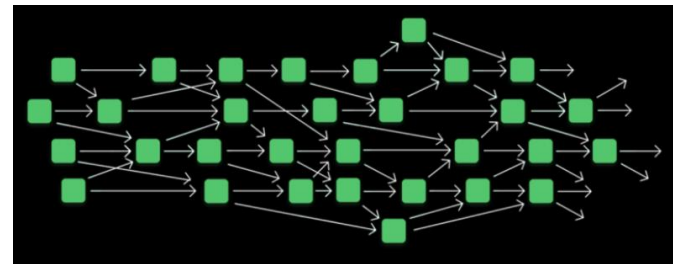
The total supply of coins is either issued to users or “pre-mined,” and consensus is done very quickly.

Consensus in DAG is **probabilistic**.

DAG has an extremely high theoretical limit on transactions per second (TPS throughput) because of the way consensus is achieved.

## Issues

- In the future, as the network expands and becomes more secure, they expect the platform to be fully decentralized.
- In other words, a **purely decentralized DAG system has not been thoroughly tested** or used as compared to blockchain.



# Directed Acyclic Graphs (DAGs)

---

DAGs are hotter than Vitalik's Tinder profile right now. DAGs are a form of consensus that doesn't use the blockchain data structure and handles transactions mostly asynchronously.

Popular Implementations: Iota, Hashgraph, Raiblocks/Nano

Pros: Network Scalability; low cost

Cons: Depends on implementation


Tangle is the DAG consensus algorithm used by Iota.

- In order to send an Iota transaction, you need to validate two previous transactions you've received. The two-for-one, pay-it-forward consensus strengthens the validity of transactions the more transactions are added to the Tangle.
- if someone can generate 1/3 of the transactions, they could convince the rest of the network their invalid transactions are valid.

Hashgraph is a gossip-protocol consensus developed by Leemon Baird.

- Nodes **share their known transactions with other nodes at random** so eventually all the transactions are gossiped around to all of the nodes.
- Hashgraph is really fast (250,000+ transactions per second) but isn't resistant to Sybil attacks.

# IOTA



**IOTA** is a distributed ledger system designed for the Internet of Things (IoT), which is used to meet the micro-payment demands between a large number of machines

**First** to use Directed Acyclic Graphs (DAG) known as **Tangle**, which overcomes the inefficiencies of current Blockchain designs and introduces a new way of reaching consensus in a decentralized peer-to-peer system. Verify 2 other transactions. (in the following slides)

The “transaction issuer” does **not pay a direct fee** for processing its own transactions – he/she only indirectly pays (with computer hashing power) by confirming other transactions

People and machines can transfer money and/or data without any transaction fees in a trustless, permissionless, and decentralized environment

With the Internet of Things, IOTA can be used for many purposes simply and automatically. For example, if a toll booth and a car both have Internet of Things sensors enabled


By July 28th, 2018, IOTA was the ninth largest crypto-currency in the world, with a total market capitalization of 27.7 billion US dollars

IOTA and Nano, indicate that they can currently process ~1,000 and 7,000 tps respectively



# What is Tangle of IOTA?

---



The **Tangle** is a DAG, where vertices represent transactions, and edges represent approvals. Considered as a **consensus method** alternative to **PoW and PoS**.

When a new transaction is introduced, it is added as a new vertex in the DAG. Verification based on 2 previous transactions using a weighted random walk.

Once a transaction is approved by a large number of newer transactions, it becomes part of the consensus and is practically impossible to alter.

The Tangle Network, in theory, doesn't have a scalability cap

The Tangle achieves this by requiring each transaction to perform a **small proof-of-work computation**. Use small PoW to **prevent spamming**. No transaction fee.

# Hashgraph

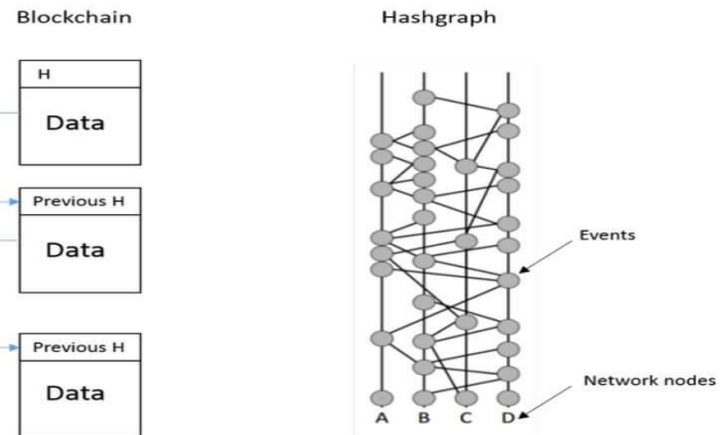
**Hashgraph** is another type of DLT and the final one to be discussed in our distributed ledger technology for dummies guide.

The idea for Hashgraph was developed by Leemon Baird in mid-2016 and was originally intended for the private corporate sector.

It is known to be **a patented technology** and is aiming to be used with permissioned blockchains.

Hashgraph has already found a potent customer who uses their **private Hashgraph software** and has even preferred it to other alternatives such as Hyperledger

Figure 18: The Structure of a Blockchain and the Hashgraph in Comparison.



Source: The Hindu<sup>114</sup>

With the Hashgraph concept, the necessary information within the network is also transferred via the so-called Gossip protocol

# Gossip protocol and Virtual Voting

---

The technique used to **share information** is called **Gossip about Gossip**.

Each member starts with a transaction, which results in an 'event'.

Then, each member **calls** another **randomly** selected member and the two share their transaction history.

For example, D calls B and shares D's transaction history with B.

This type of call happens repeatedly, with each member randomly calling another member and sharing its transaction history

The **second technique of the hashgraph** is **Virtual Voting**, and its purpose is to reach a consensus on the order of transactions.

First, the events are divided into rounds. The **hashgraph algorithm** has a definite mathematical answer for when a round is created

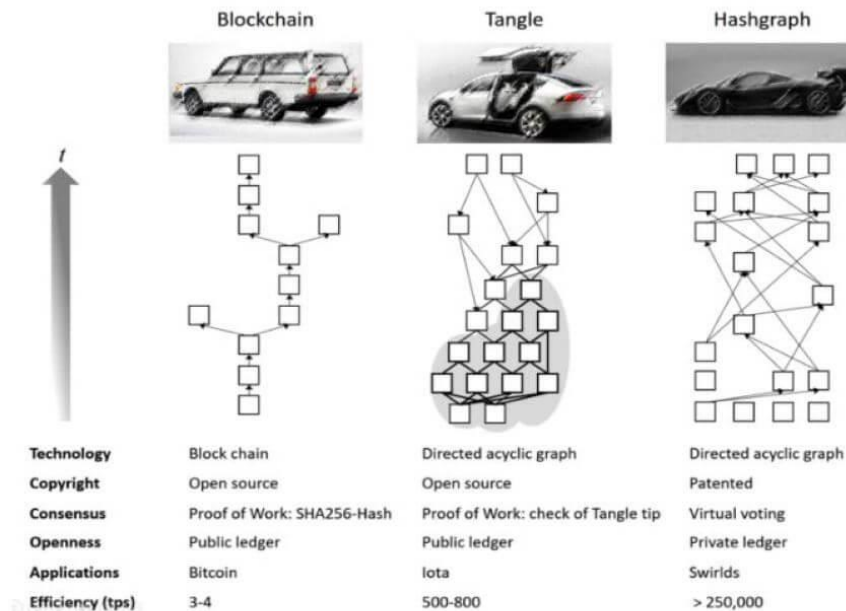
Each member **votes to determine** which event should qualify as a 'famous witness'.

To understand how this happens, imagine that each of the members with an event in the next round looks backwards to each event in the current round to see if it can trace its lineage back to the current round's event.

If it can trace its lineage back to an event, it votes yes for that event, and if not, it votes no.

# Comparison of various DLT

Figure 19: A Simple Overview of the Most Important Key Points of the Various Distributed Ledger Technologies.



Source: Fintech News<sup>116</sup>

# Spectrum of Blockchain and DLT

---

Andreas Wallendahl of ConsenSys explains this nicely by placing blockchain and distributed ledgers on a spectrum. On the one side, you have public blockchains. These have many nodes, are decentralized and can be used by anyone in the world.

At the other end, you have a single database with one node, and next to that, as you move a little toward the middle of the spectrum, a database shared between just two or three nodes or a distributed ledger.

# Two Issues with Blockchain technology

---

1

## Scalability Issue

- Since each node on the network must verify each transaction, the network becomes slower with more transactions.
- For example, if 1,000 transactions are broadcasted to the network at once, then each node (or a large majority) will ultimately need to agree that each transaction is valid.
- Substantial scaling
- Limited in terms of transaction throughput and speed
  - Bitcoin – 7 transactions per second
  - Ethereum – 20 transactions per second

2

## High fee Issue

- With few miners, these 1,000 transactions can take hours to complete, and only those with high fees will gain higher priority.

# Comparison of various DLT

	Blockchain	Sidechain	Tangle	Hashgraph
Data Structure	Linked list	List of linked lists	DAG	DAG
Consensus	PoW: SHA256-Hash	PoW: Ethash	PoW: hashcash	Virtual voting
Transactions	Grouped into blocks	Two chains of blocks	Single transactions	Gossip events contains transactions
Fees	Yes	Yes for the public chain	No fee	No fee
Tps	4 to 7	Limited by consortium chain	500 to 800	> 200,000
Validation time	Order of minutes	Order of minutes	Order of seconds	Order of seconds
Privacy	Low	High	Low	Low
Security	High	High	High	High
Maturity	Many implementation	Experimental	Experimental	Experimental
Platforms	Bitcoin, Ethereum	Ethereum and Monax	IOTA	Hedra
Copyright	Open source	Open source	Open source	Patented
Typologies	Public	Public and private	Private	Private

# Comparison of different DLTs

SWOT	Blockchain	Sidechain	Tangle	Hashgraph
Strengths	<ul style="list-style-type: none"> <li>- Transparency</li> <li>- Many implementations</li> <li>- It is commercially in use</li> </ul>	<ul style="list-style-type: none"> <li>- High privacy - Performance</li> </ul>	<ul style="list-style-type: none"> <li>- No mining needed</li> <li>- No transaction fee</li> <li>- Quantum security</li> </ul>	<ul style="list-style-type: none"> <li>- High number of transactions</li> <li>- Fairness</li> <li>- Scale in the number of transactions</li> </ul>
Weaknesses	<ul style="list-style-type: none"> <li>- Small block size</li> <li>- Low processing speed</li> <li>- Low scalability</li> <li>- Lack of interoperability</li> <li>- Transactions fees</li> </ul>	<ul style="list-style-type: none"> <li>- Permissioned and private</li> </ul>	<ul style="list-style-type: none"> <li>- Managed by single organization</li> </ul>	<ul style="list-style-type: none"> <li>- Gossiping overhead</li> <li>- employs coin toss to terminate consensus</li> <li>- Permissioned network</li> </ul>
Opportunities	<ul style="list-style-type: none"> <li>- Automation</li> <li>- Traceability</li> <li>- M2M interaction</li> <li>- world wild fast transactions</li> </ul>	<ul style="list-style-type: none"> <li>- M2M interaction</li> </ul>	<ul style="list-style-type: none"> <li>- Machine economy</li> </ul>	<ul style="list-style-type: none"> <li>- Innovation In Almost Every Industry Especially payment</li> </ul>
Threats	<ul style="list-style-type: none"> <li>- Regulation might be a problem - legal compliance</li> <li>- Gov. Willingness to adopt</li> </ul>	<ul style="list-style-type: none"> <li>- Interoperability between platforms</li> <li>- still experimental</li> </ul>	<ul style="list-style-type: none"> <li>- More research is needed - still experimental</li> </ul>	<ul style="list-style-type: none"> <li>- Patented technology</li> <li>- More research is needed</li> <li>- Works in a permissioned setting</li> <li>- making sure that each node is connected to at least one honest node is and open issue</li> <li>- Governing Council</li> </ul>



# Comparison of DAG and Blockchain


---

## SIMILARITIES

Both use public/private-key cryptography

Both use peer-to-peer model for communication

Both need to have consensus algorithm



## DIFFERENCES

Blockchain based on linked block;  
DAG based on tree graph

Some public Blockchain uses native currency, which is a mandate; DAG does not

Blockchain transactions needs to be grouped into blocks before transaction updated to the chain; DAG does not

Excerpt From: Debajani Mohanty. "R3 Corda for Architects and Developers". Apple Books.

# Pros and Cons of DAG

---

## PROS

- More decentralized when compared to blockchain
- Cut down on operational inefficiencies with less overhead
- Faster
- Better scalability due to DAG data structure

## CONS

- Also vulnerable to double-spend attacks
- Network synchronization of status is probabilistically achieved

# Corda of R3

---

**R3 Corda** is a **distributed ledger platform** developed independently by R3REV, which is a Blockchain consortium of more than 200 of world's biggest banks and financial institutions consisting of R3. R3 was a financial services technology company founded in 2014 by David Rutter and leads a consortium of 200+ members including Microsoft and Intel

More than 50 global financial institutions such as Goldman Sachs, JP Morgan, Morgan Stanley, and USB are participating in R3 Corda.

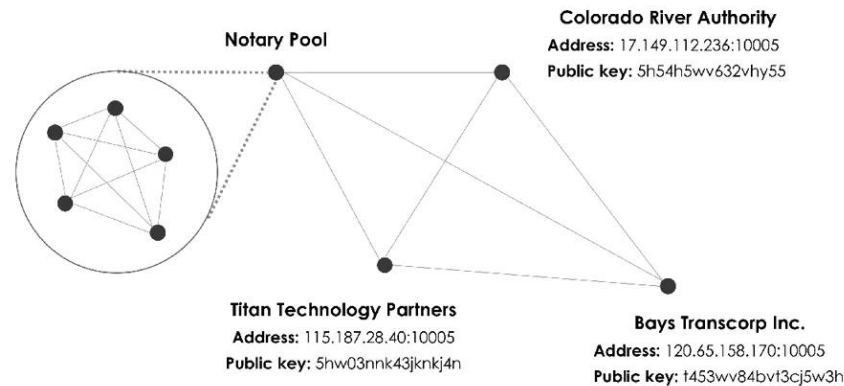
It was developed and launched by R3 in 2016. R3 is an enterprise software firm which focuses on distributed ledger technology

Another enterprise ready blockchain platform other than Hyperledger

Corda is a **distributed database technology** that equally records/archives the evolution of Contract State (state of the deal) through trading partners' consensus.

Corda creates consensus of a transaction at a deal level between trading partners and divides the consensus into validity and uniqueness

# Corda R3



- Private but authenticated/authorized connected parties (Permissioned and semi-private)
- Corda restricts access to data within an agreement to only those explicitly entitled to it
- **No native currency**
- Open source blockchain project
- With **smart contracts** that facilitate direct transactions among the businesses.
- With authenticated peer-to-peer network of nodes.
- Each node is a JVM run-time environment hosting Corda services and executing applications known as CorDapps.
- Supported DLT but block-less platform

# Corda Architecture

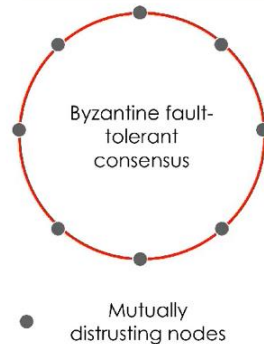
Building blocks of Corda such as its network service for managing the **registration process of nodes**

In the Corda network, **each node is a verified IP address** that is agreed between business counterparties before finalizing a contract.

These IP addresses representing different organizations go through a stringent KYC process and are added to the network through a network map service called “the doorman”.

In Corda,

- each node maintains a **separate database of data** that is only relevant to it
- **no concept of global broadcasting of data**, and all communications are only on peer-to-peer basis.
- does **not come with blocks** and it is not a Blockchain
- Perform with real-time synchronous transaction



Transaction Hash	Output Index
622B2C606A23FEAE2798170	0
CC739CA364D7B396FE960B	2
628B4CE58B4A5764948AC4E	1
18265993CBC000D12DCCEE	4
F0F01F2ED2B442452499567E	2
...	...

# Corda R3 Ledger

---

In Corda, when nodes wish to do a business transaction with each other, the data is stored in their individual databases or ledgers after contractual validation and verification by notary

Identity in Corda is related to the identity of nodes that actually stand for the individual organizations that participate in the DLT network.

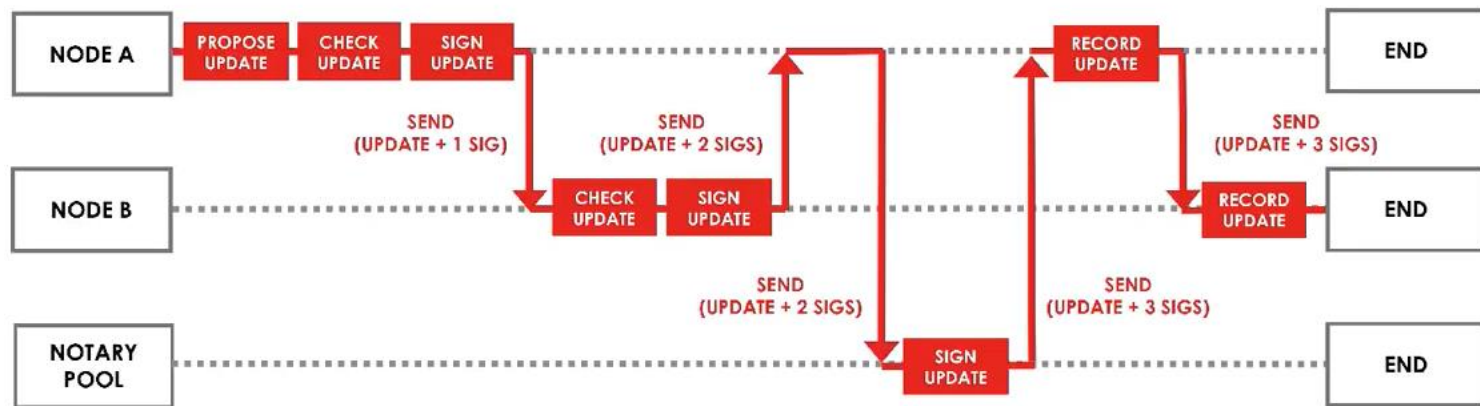
In Corda, data is **maintained by state objects**.

In Corda, every transaction to be executed has to be **signed by each of the concerned nodes' notaries** and also has to be contractually valid.

Smart contracts in Corda can be written in JVM languages as Kotlin or Java

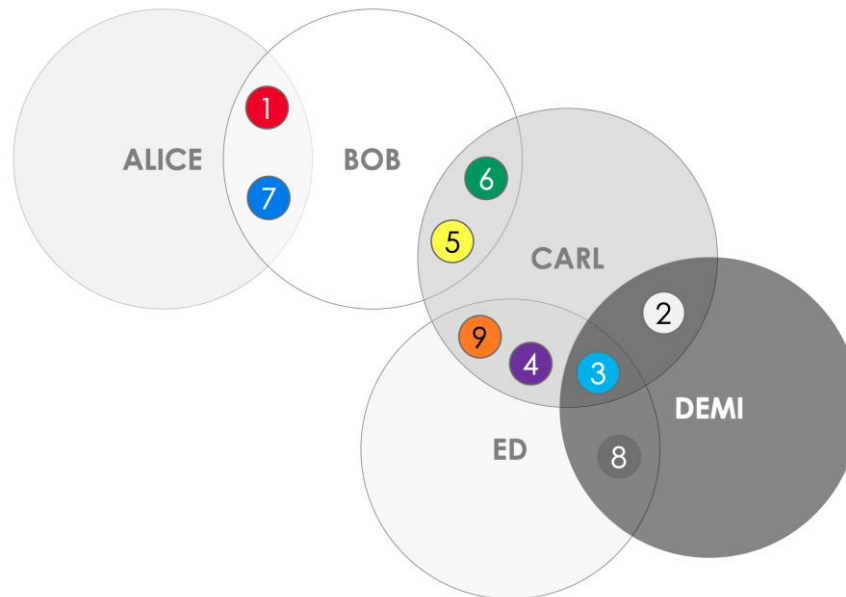
Corda defines a transaction as “a proposal to update the ledger” that will be committed if contractually valid, that is, signed by all relevant parties and notary

# Corda R3 Transaction



# Corda R3 Ledger

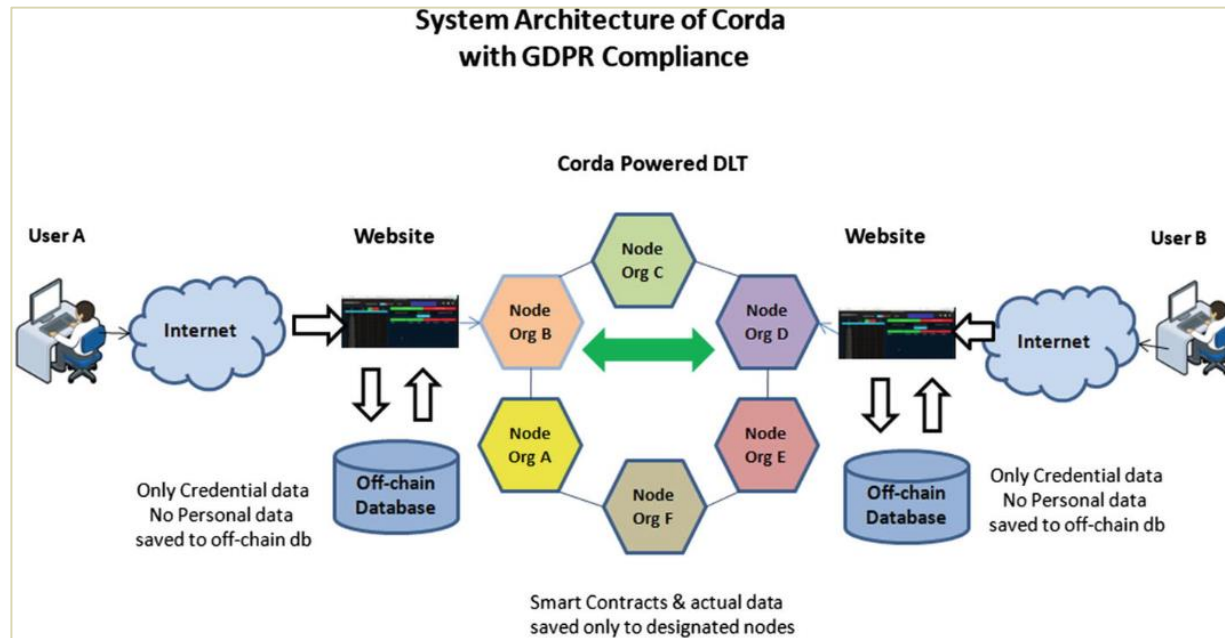
---



Corda guarantees that whenever one of these facts is shared by multiple nodes on the network, it evolves in lockstep in the database of every node that is aware of it



# Corda's Compliance with GDPR



Excerpt From: Debajani Mohanty. "R3 Corda for Architects and Developers". Apple Books.

# Comparison of Private/Consortium blockchain platform

	Hyperledger Fabric	Quorum	Ethereum	Ripple	R3 Corda
Ledger Type	Permissioned	Permissioned	Permissionless	Permissioned	Permissioned
Governance	Linux Foundation	JP Morgan and Ethereum developers	Ethereum developers	Ripple Labs	R3 Consortium
Industry-focus	Cross-Industry	Multi-Industry	Cross-Industry	Financial Industry	Financial Industry
Throughput	> 2000 tps	100 tps	~ 20 tps	~ 1500 tps	~170 tps
Cryptocurrency	None	None	Ether (ETH)	Ripple (XRP)	None
Consensus Mechanism	Pluggable Mechanism	Voting Protocol	Proof of Work (PoW)	Probabilistic Voting Protocol	Pluggable Mechanism
Smart Contract	✓	✓	✓	×	✓
Smart Contract Language	NodeJS or Golang or Java	Solidity	Solidity	–	Java or Kotlin
Application Type	Wide-ranging	Broad-ranging	Wide-ranging	Suited for financial applications	Financial applications

# Lab References

---

# Setup and implementation of Blockchain, HyperLedger

---

<https://aws.amazon.com/blogs/database/build-and-deploy-an-application-for-hyperledger-fabric-on-amazon-managed-blockchain/>

<https://hackernoon.com/hyperledger-fabric-installation-guide-74065855eca9>

<https://medium.com/@eSizeDave/https-medium-com-esizedave-how-to-install-hyperledger-fabric-1-2-on-ubuntu-16-04-lts-ecd4d4cec72>

<http://www.ziaahmedshaikh.com/install-hyperledger-fabric-on-ubuntu-18-04-1-step-by-step/>

<https://github.com/aws-samples/hyperledger-on-kubernetes/blob/master/fabric-main/README.md>

<https://github.com/aws-samples/non-profit-blockchain/blob/master/ngo-fabric/README.md>

<https://medium.com/coinmonks/get-started-with-blockchain-using-the-aws-hyperledger-fabric-template-an-unofficial-guide-551bc46af710>

FabCar exercise

- <https://medium.com/@mycoralhealth/build-a-dapp-on-hyperledger-the-easy-way-178c39e503fa>
- [https://hyperledger-fabric.readthedocs.io/en/release-1.4/write\\_first\\_app.html](https://hyperledger-fabric.readthedocs.io/en/release-1.4/write_first_app.html)