# Concepts & Architecture

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

**Discount Coupon Links to UDEMY courses:**

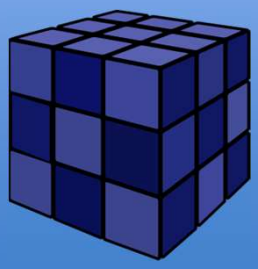
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099



https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on "Hyperledger Fabric Development with Composer"

# Hyperledger Concepts

raj@acloudfan.com

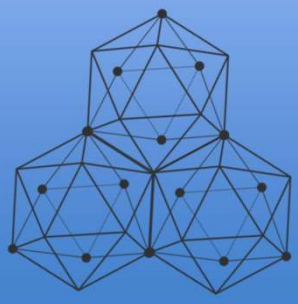
@acloudfan

http://ACloudFan.com

**Learning Objectives:**

http://www.ACloudFan.c

- Assets
- Chaincode
- Ledger



## Assets

- JSON or Binary



```
{
    vinNumber: …
    owner: ….
}
```

- Transactions = Change of Asset's state

## Assets

- JSON or Binary



```
{
    vinNumber: …
    owner: ….
}
```

## Chain code

- Defines the asset's structure
- Transaction | Business logic

Changes State of Asset

*Function sellTheCar(newOwner)*

## Ledger

- Tracks all of the asset Transactions
  - Records State changes of the asset



- Ledger is distributed (DLT)
  - All participants have a replica of the ledger

# Summary

Fabric
HYPERLEDGER

- Represents value

**Assets**



- Adds transaction
- Record state

**Chain code**

- Defines structure
- Code the transactions

**Ledger**

# Hyperledger Concepts

raj@acloudfan.com

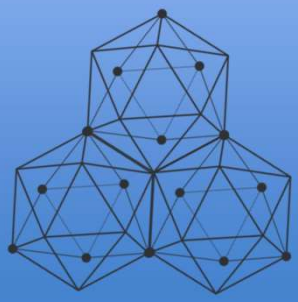
@acloudfan

http://ACloudFan.com

**Learning Objectives:**



- Permissioned network
  - Identities
  - Membership Service Provider



## Businesses interact with known entities

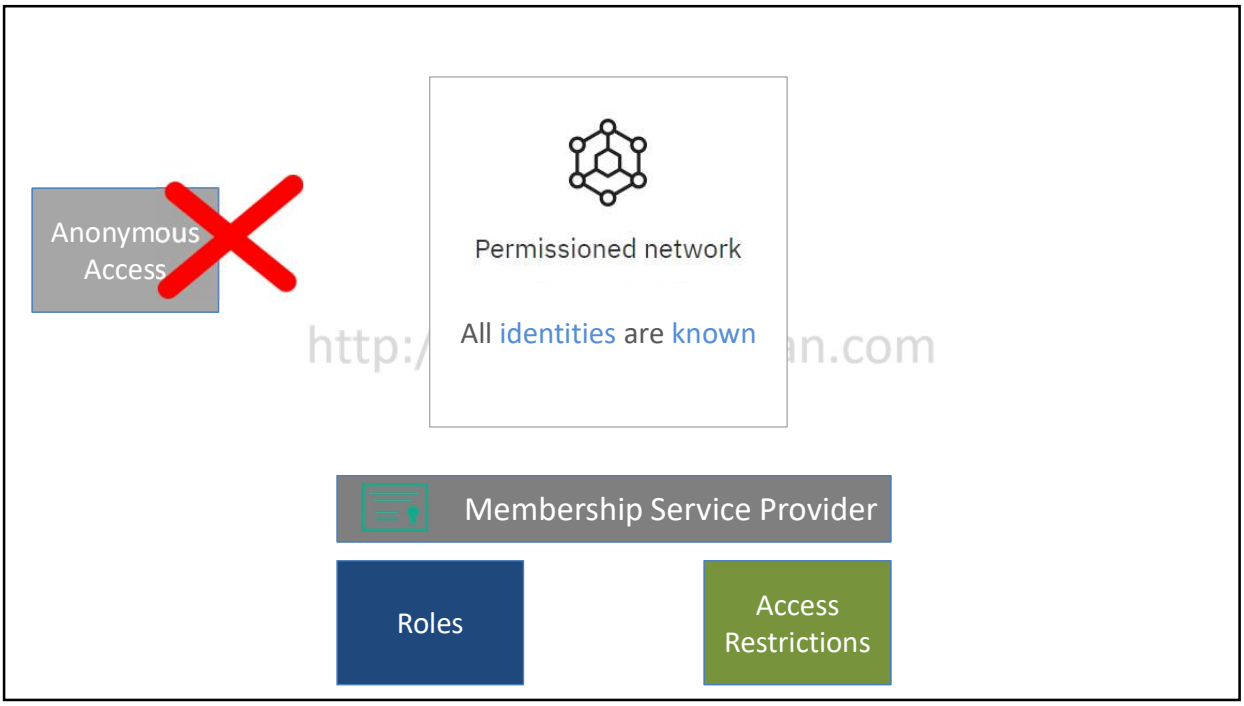


ACME Corp.

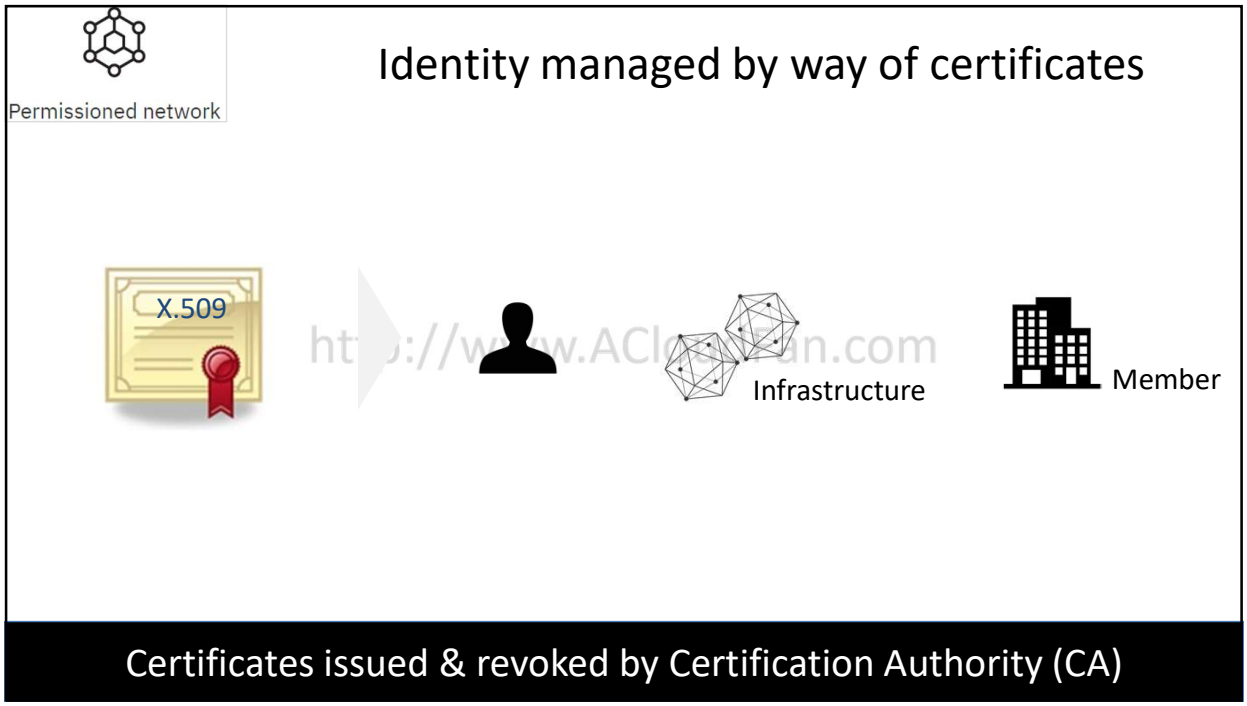
ABC Corp.

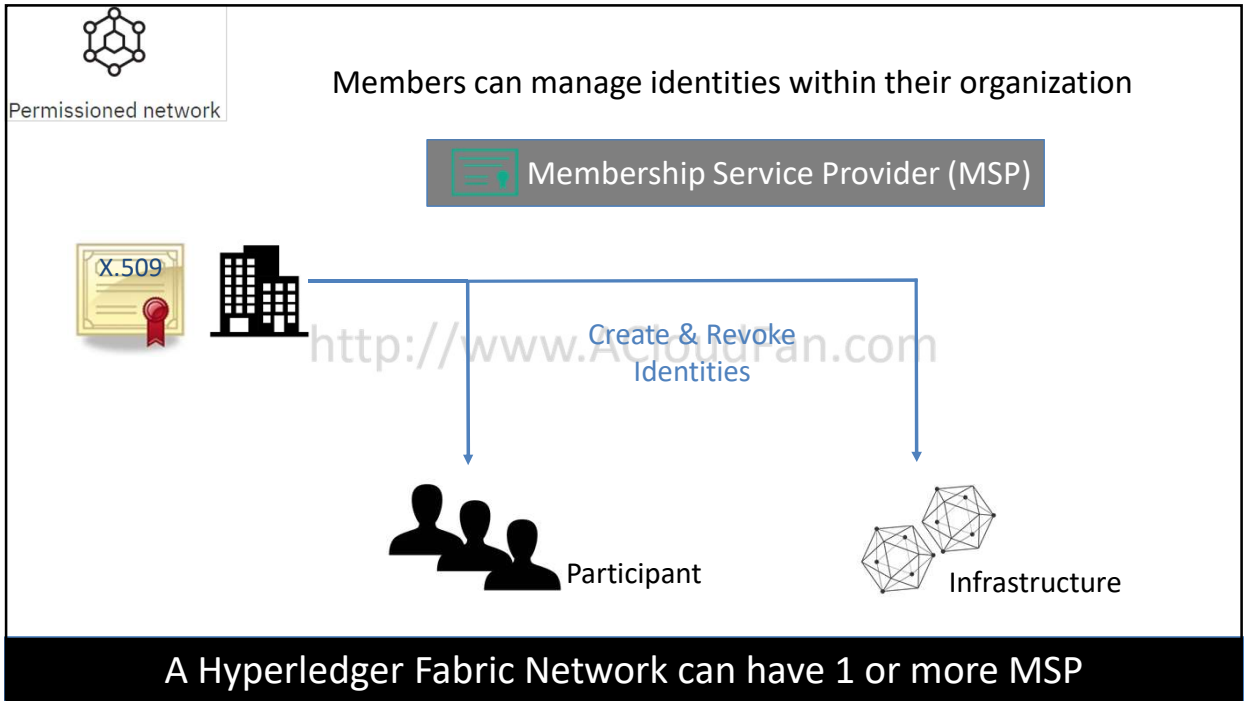
John Doe

Regulatory


Anonymous Access
Permissioned network
All identities are known
Membership Service Provider
Roles
Access Restrictions


Member = Legally separate entities
Manufacturer
A
B
C
Dealer
Repair Shop


Permissioned network
Identity managed by way of certificates
X.509
Infrastructure
Member
Certificates issued & revoked by Certification Authority (CA)


Permissioned network
Members can manage identities within their organization
Membership Service Provider (MSP)
X.509
Create & Revoke
Identities
Participant
Infrastructure
A Hyperledger Fabric Network can have 1 or more MSP

## Membership Service Provider

- Abstract component that manages identities
- Provides the credentials to various entities

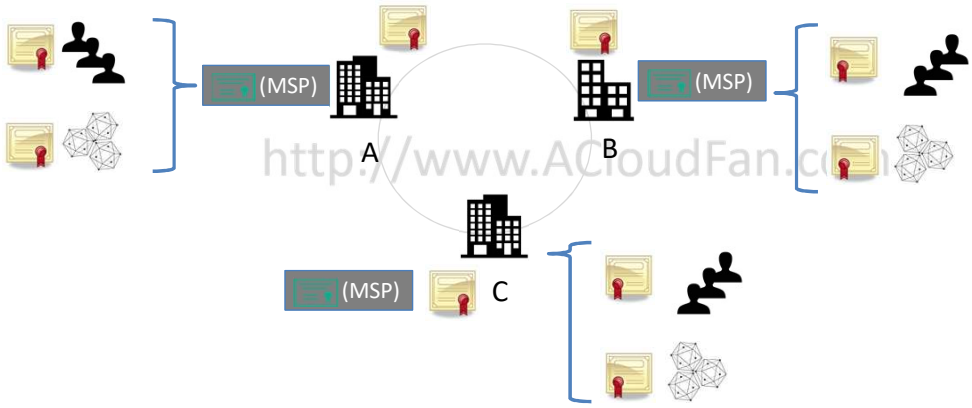


Node

- Pluggable
  - Network may have 1 or more MSP

# Summary

Permissioned network






- Members = separate legal entities
- Certificate assigned to participants, member, infrastructure

# Hyperledger Concepts

raj@acloudfan.com

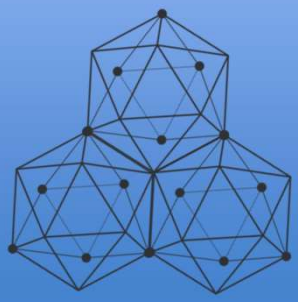
@acloudfan

http://ACloudFan.com

**Learning Objectives:**



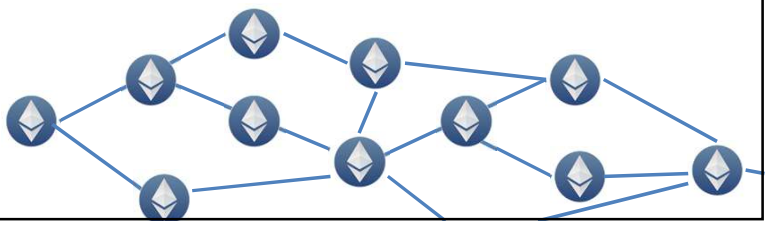
- Nodes
- Channels

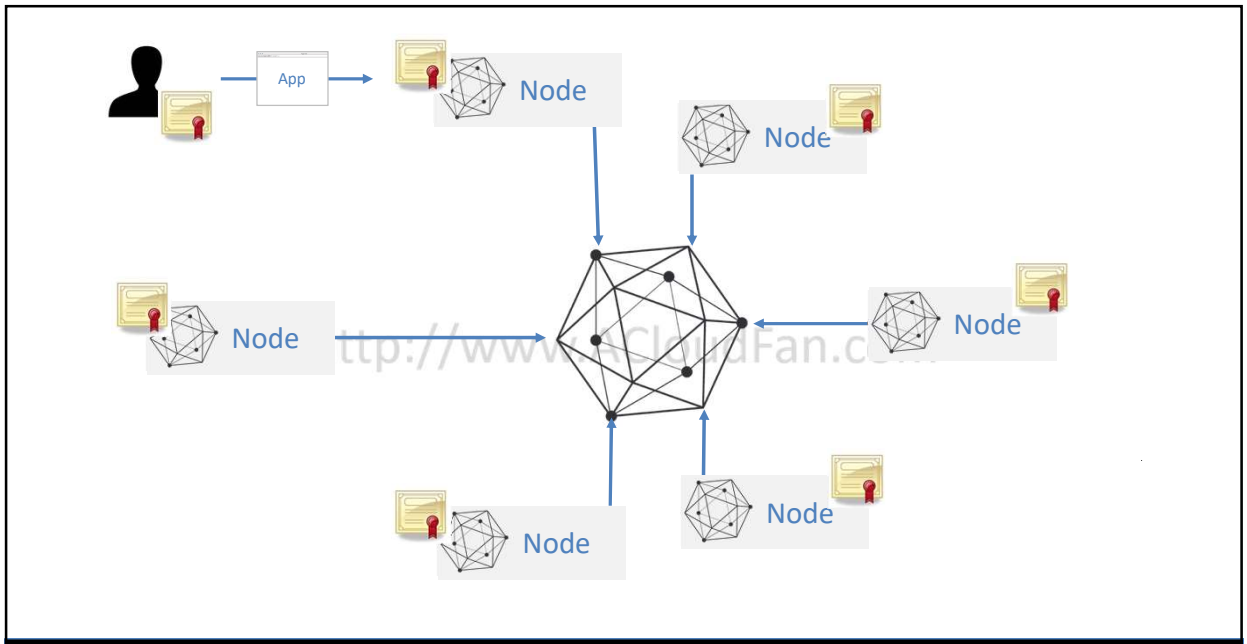


- All BC technologies have the concept of nodes
  - Nodes connect to other nodes to form the BC network

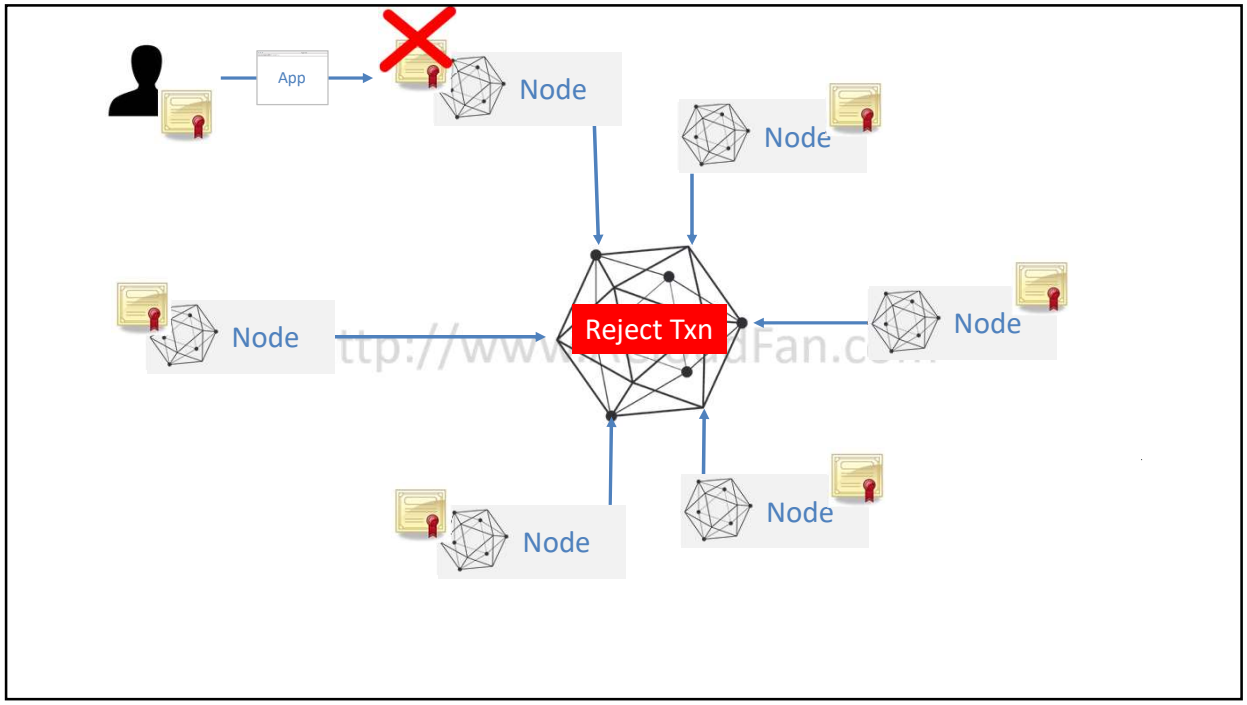


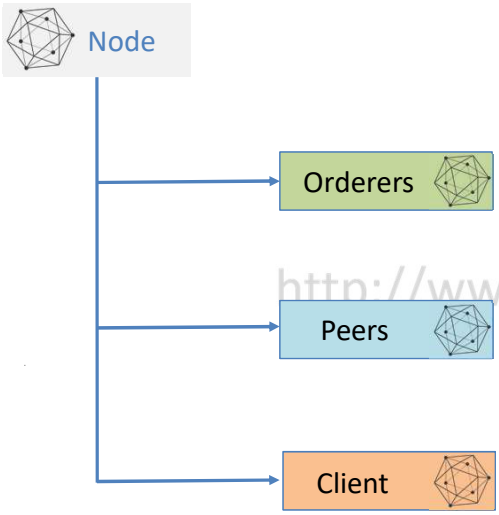
ETHEREUM bitcoin All Nodes are Equal






Node = Communication entities of the Blockchain





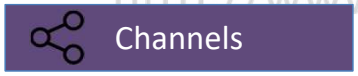

- Responsible for distribution of transactions
- Keeps the Ledger data in synch across network
- Initiates the transactions



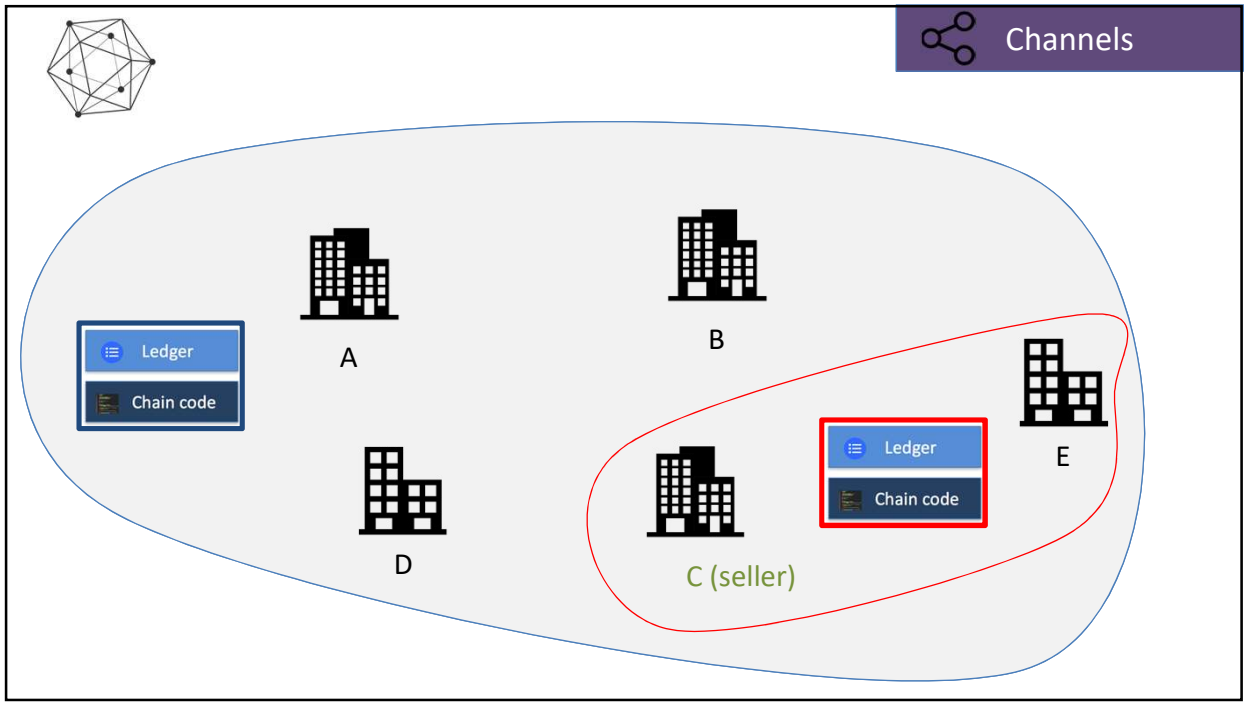
In Hyperledger Fabric = All nodes are **NOT** equal

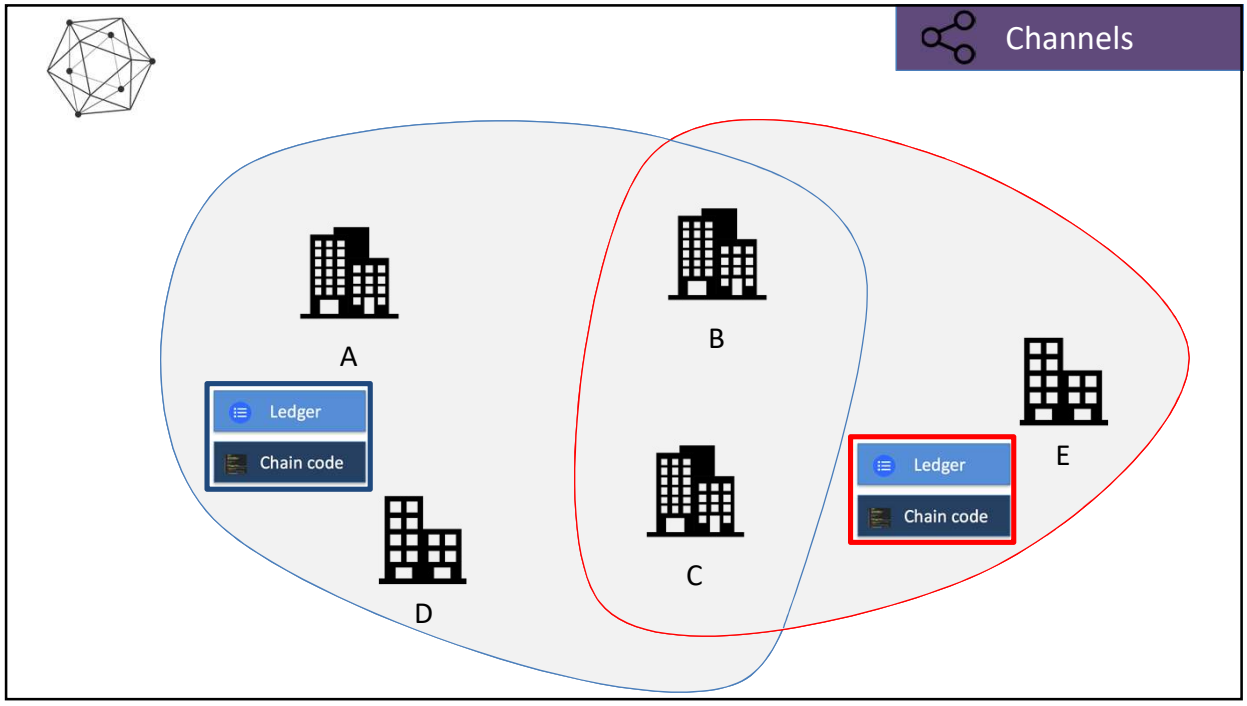
- Members can participate on multiple HL BC Networks
  - Transactions in each network is isolated






- Peers connect to the channel
- Independent Ledger in each channel


Channels
Ledger
Chain code
A
B
D
C (seller)
Ledger
Chain code
E


Channels
A
Ledger
Chain code
D
B
C
Ledger
Chain code
E

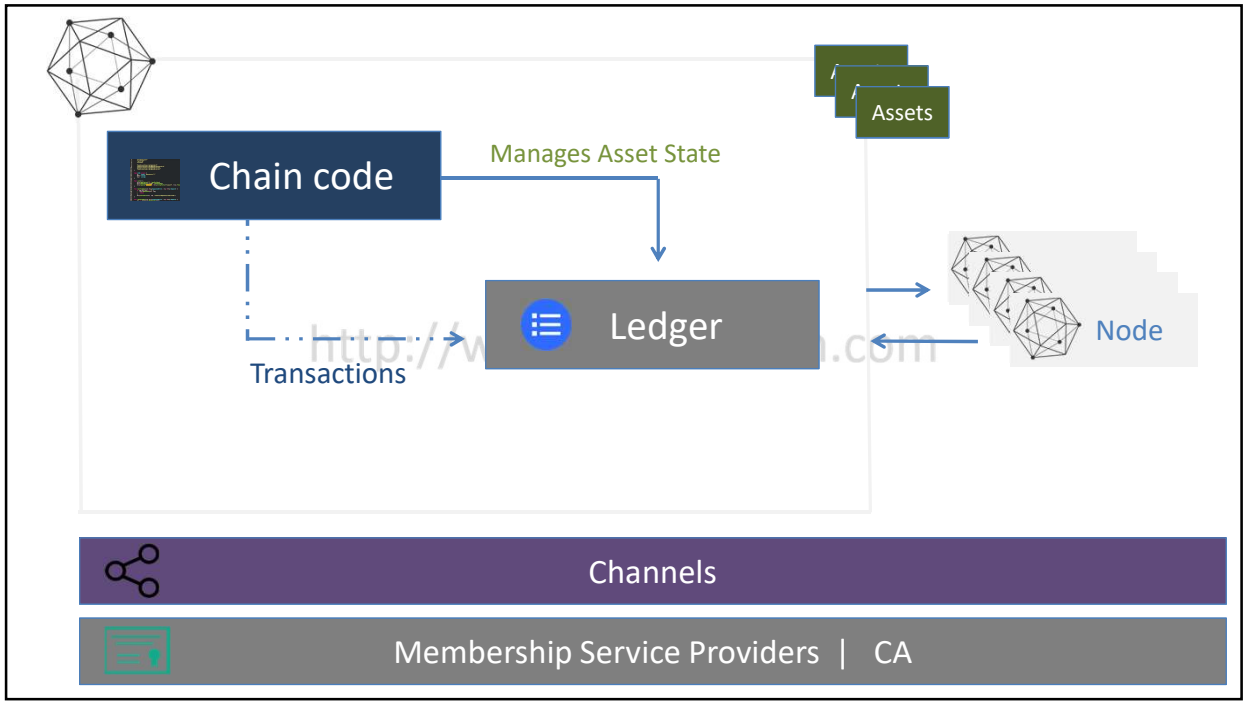
privacy

1. By way of Private channel
   - Isolates the Ledger | Transactions



2. Intermediate solution
   - Common channel
   - Chaincode installed on peers that need visibility

+ Use Data Encryption



# Summary

Node

- All nodes are NOT equal on HL Fabric

Client | Peers | Orderers



- Members participate on multiple HL BC network by way of

Channels

- Each channel manage its own independent Ledger

# Consensus

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

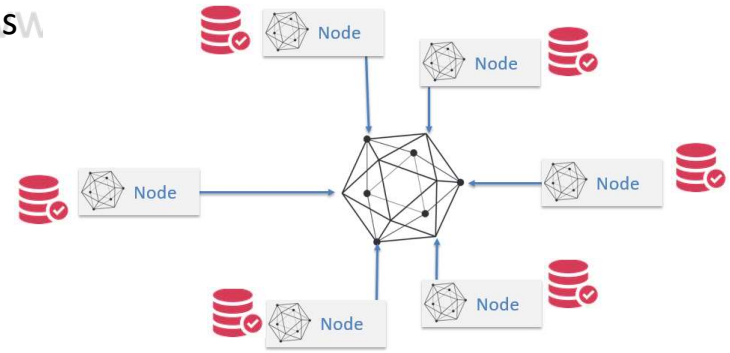
**Learning Objectives:**



- High Level Overview
  - Chaincode
  - State
  - Ledger

## Consensus

- Ensures consistency of ledger data across nodes
  - All approvers agree to the transaction
- Order of transactions




## Consensus

- Implemented as node referred to as Ordering Service
- Pluggable
  - Members in the network decide
  - New models easy to implement



- No concept of mining
- No need to incentivize
- No crypto token

# Ledger Implementation

raj@acloudfan.com

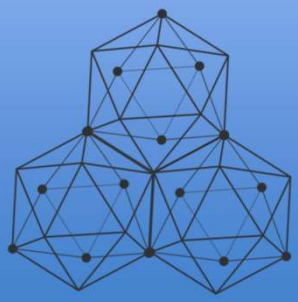
@acloudfan

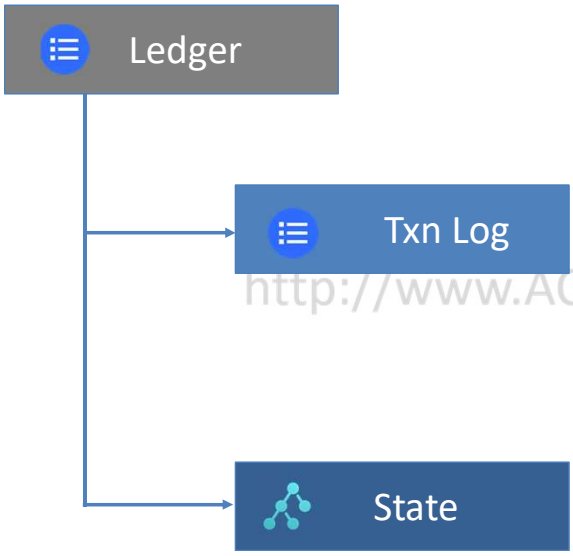
http://ACloudFan.com

**Learning Objectives:**



- Transaction Log
- State database





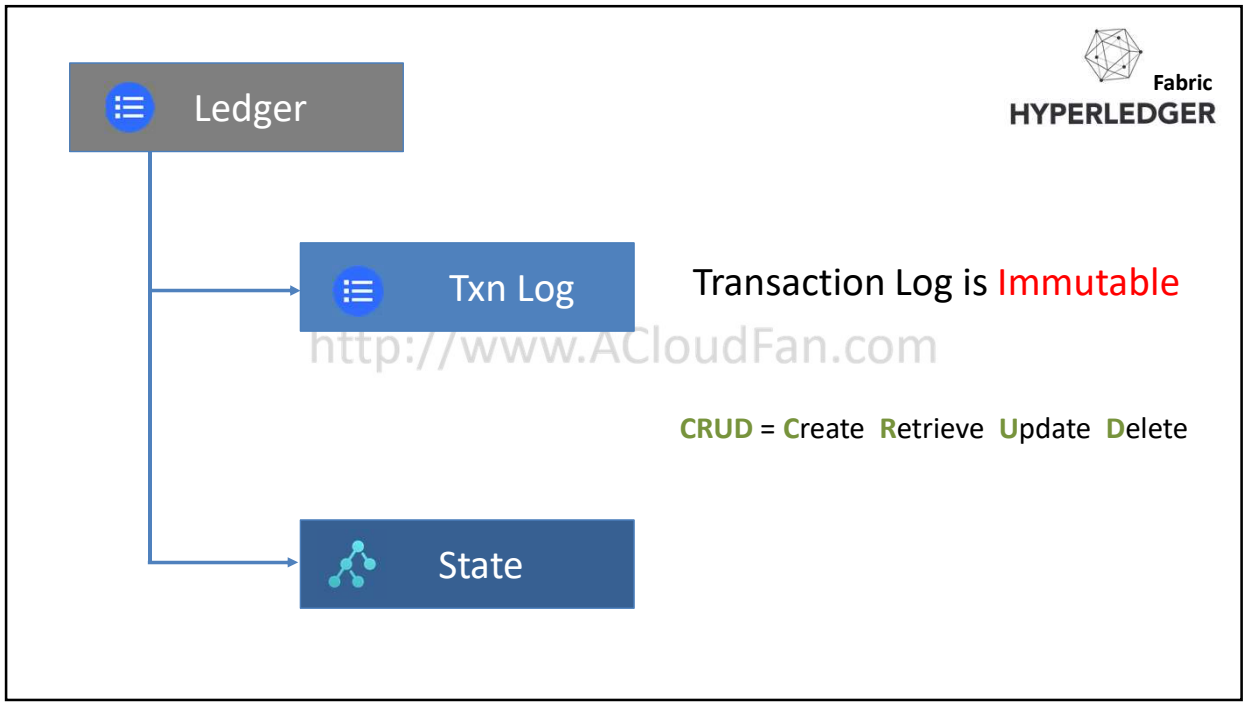

Fabric
HYPERLEDGER

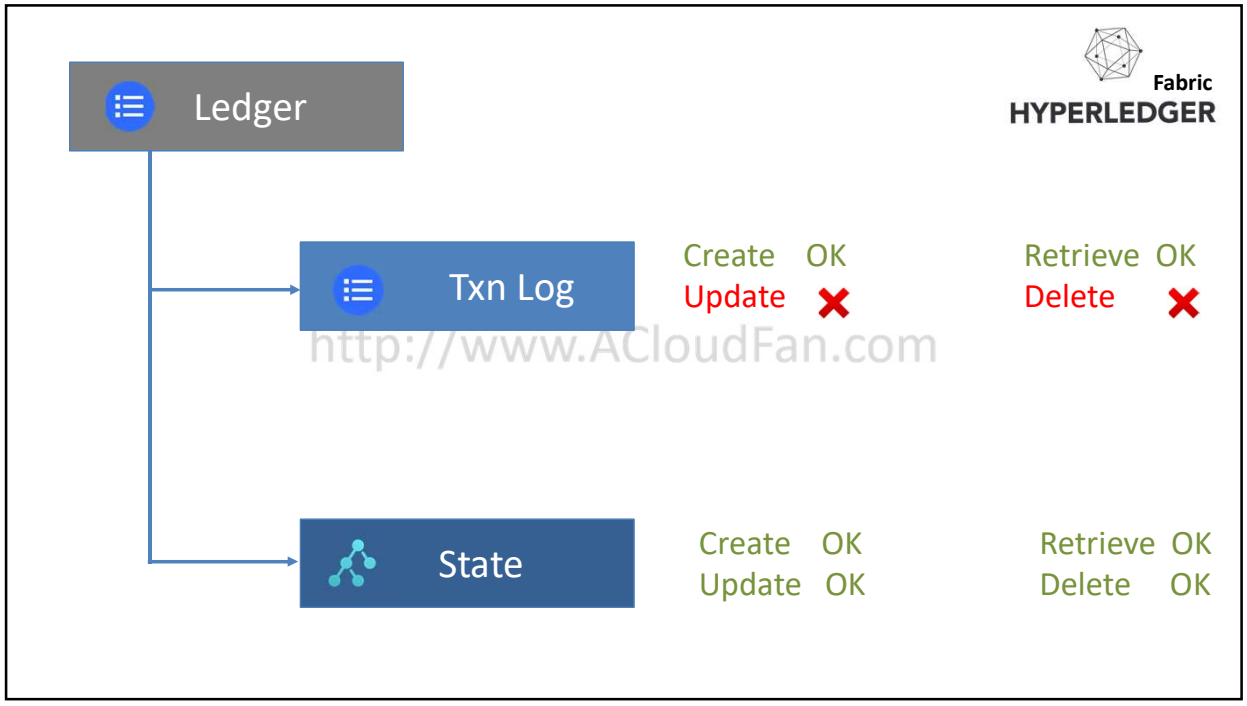
- All Transactions
  1. Change Car owner to A
  2. Change Car owner to B
  3. Change Car owner to C

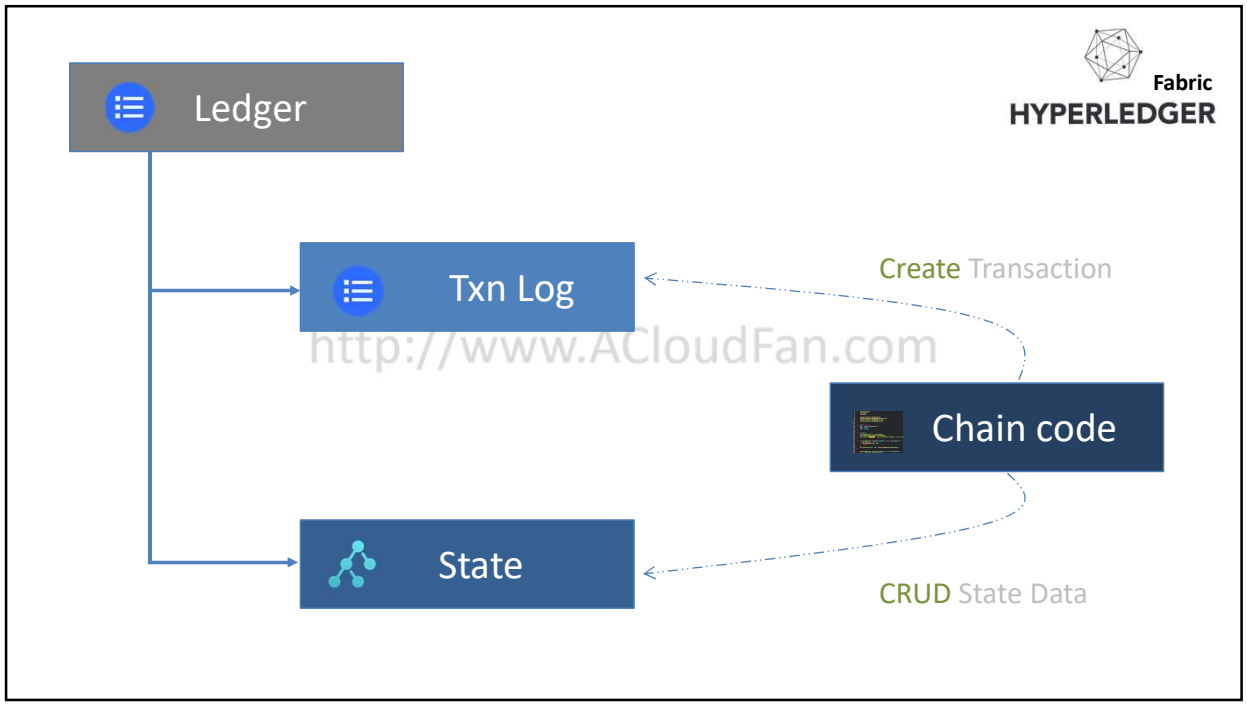


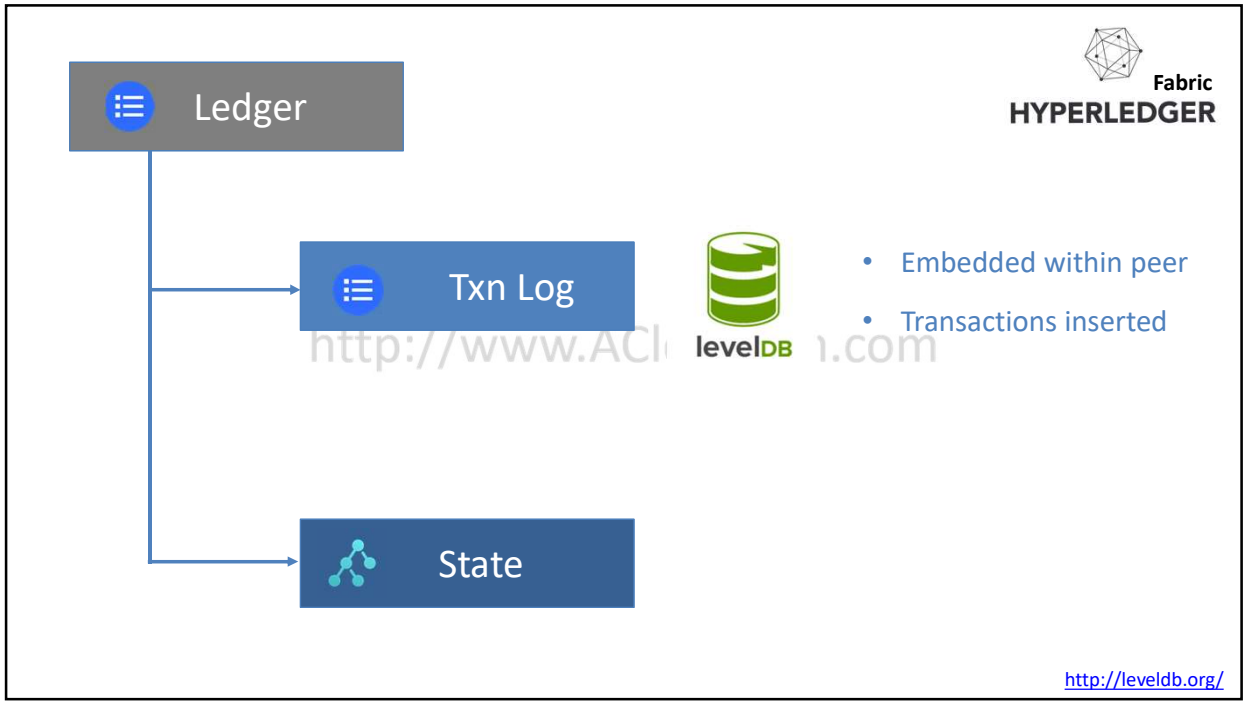
- Current state of assets

  { vinNumber: 123, owner: C }


Ledger
Fabric
HYPERLEDGER
Txn Log
Transaction Log is Immutable
http://www.ACloudFan.com
CRUD = Create Retrieve Update Delete
State


Ledger
Fabric
HYPERLEDGER
Txn Log
Create OK
Update
Retrieve OK
Delete
http://www.ACloudFan.com
State
Create OK
Update OK
Retrieve OK
Delete OK


Ledger
Fabric
HYPERLEDGER
Txn Log
Create Transaction
http://www.ACloudFan.com
Chain code
State
CRUD State Data


Ledger
Fabric
HYPERLEDGER
Txn Log
levelDB
• Embedded within peer
• Transactions inserted
State
http://leveldb.org/

## State

- Versioned

Key – Value store

New version created on state updates

Arbitrary BLOB

{JSON}



Name of State variables

Key = "Owner"


Version = 1 , 2 , 3, ...

## State

State( Key ) = (Version, Value)

- By default State Data is managed in LevelDB

Chain code

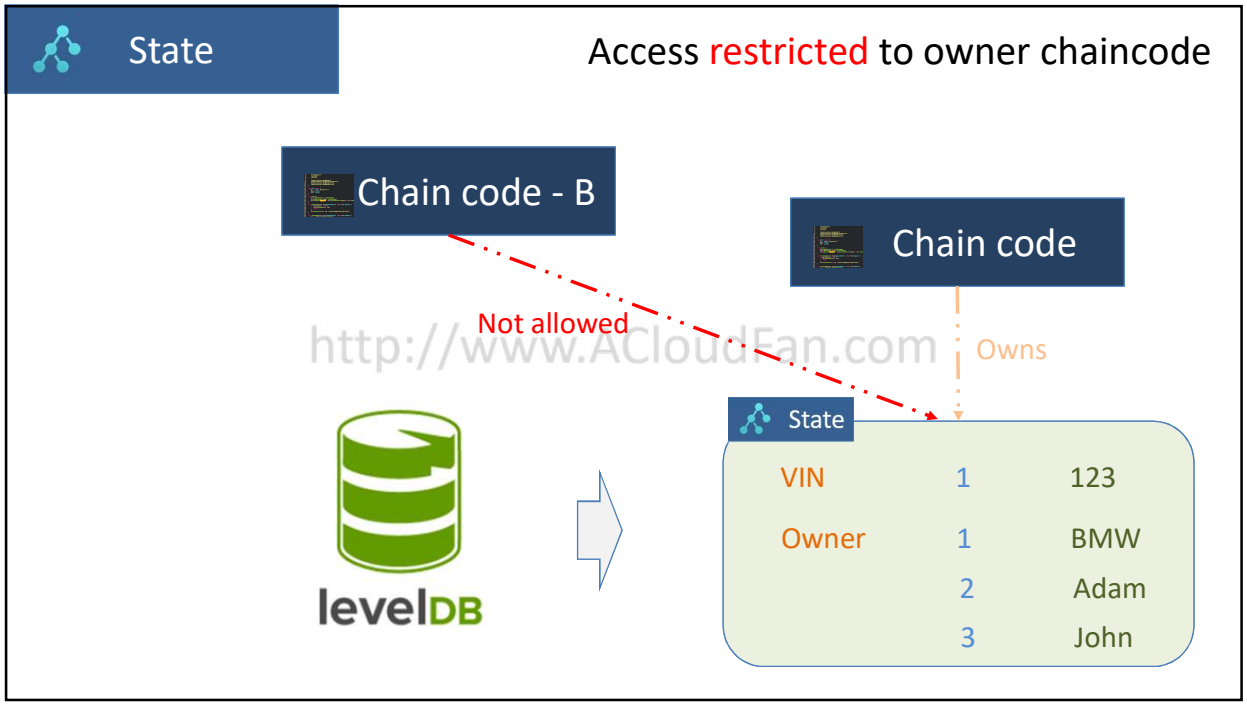


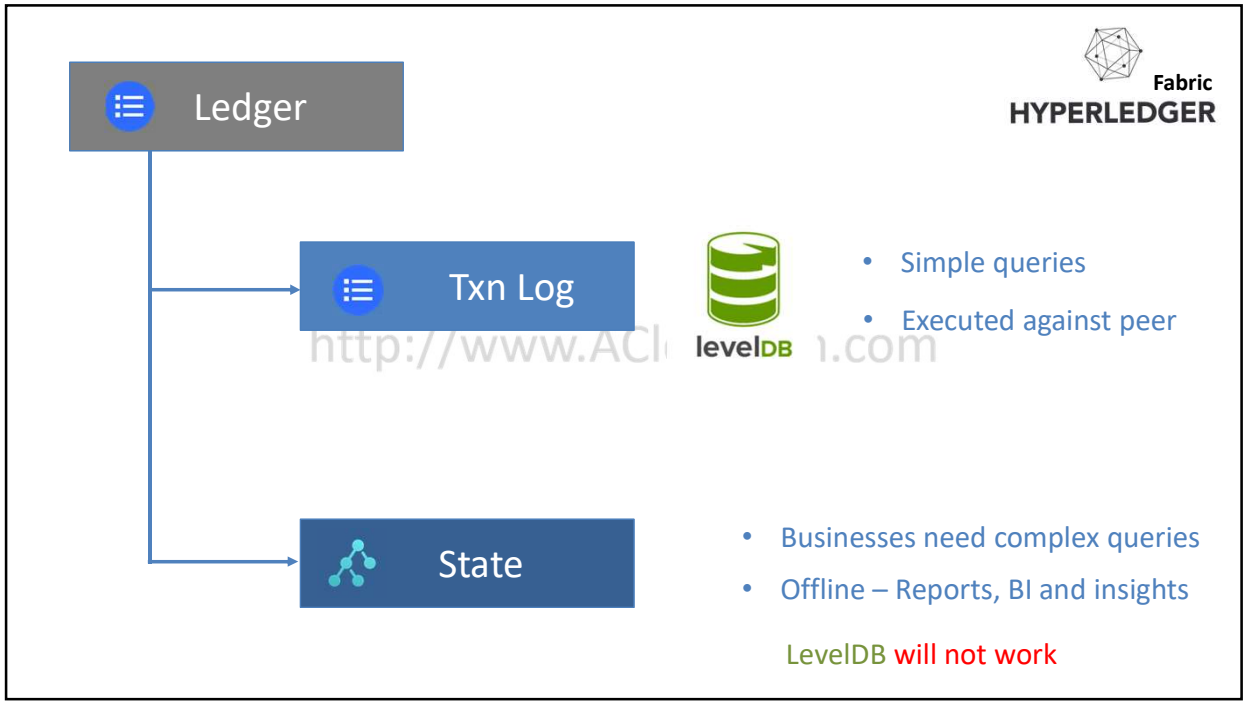
Owns

levelDB

State

| | | |
|---|---|---|
| VIN | 1 | 123 |
| Owner | 1 | BMW |
| | 2 | Adam |
| | 3 | John |


State
Access restricted to owner chaincode
Chain code - B
Chain code
Not allowed
Owns
http://www.ACloudFan.com
levelDB
State
VIN 1 123
Owner 1 BMW
2 Adam
3 John


Ledger
Fabric
HYPERLEDGER
Txn Log
levelDB
Simple queries
Executed against peer
State
Businesses need complex queries
Offline – Reports, BI and insights
LevelDB will not work

## State

- State database is Pluggable

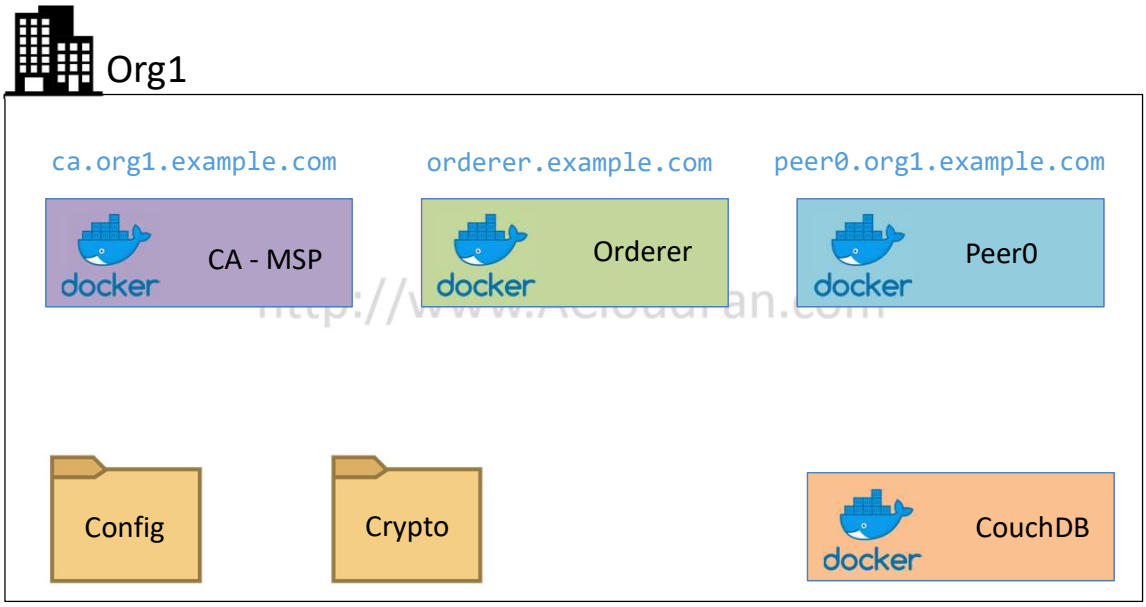
levelDB

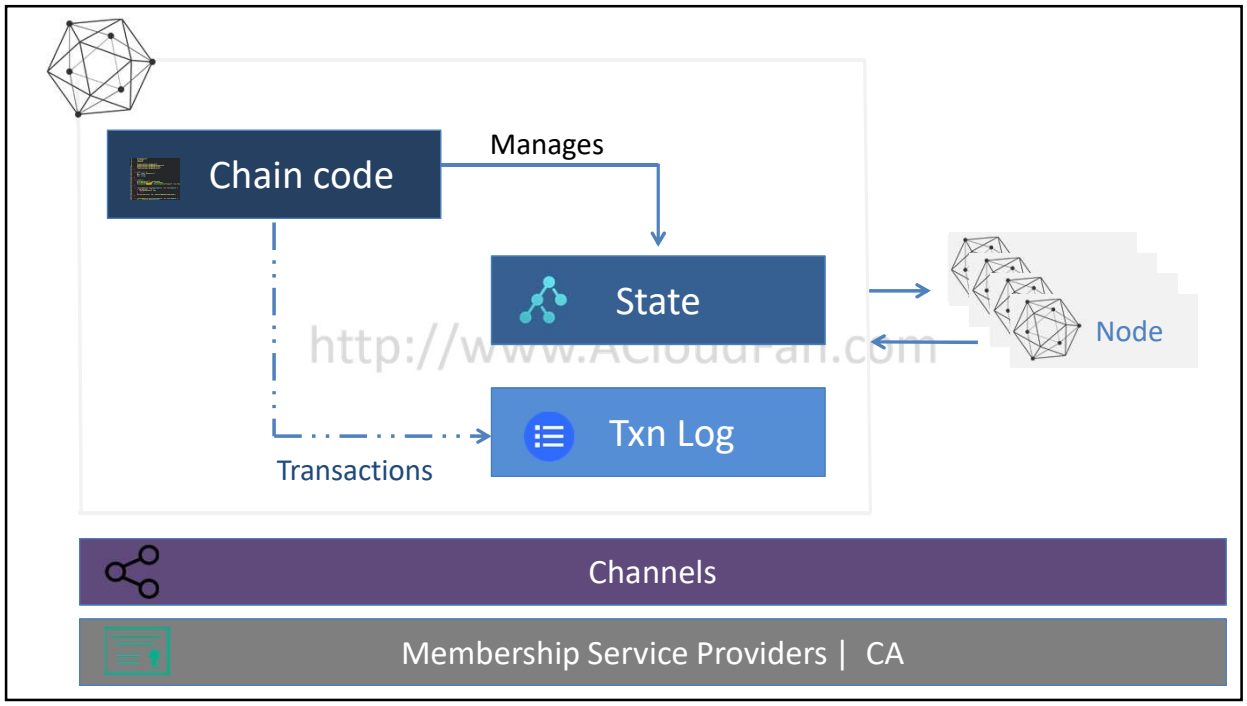


CouchDB relax

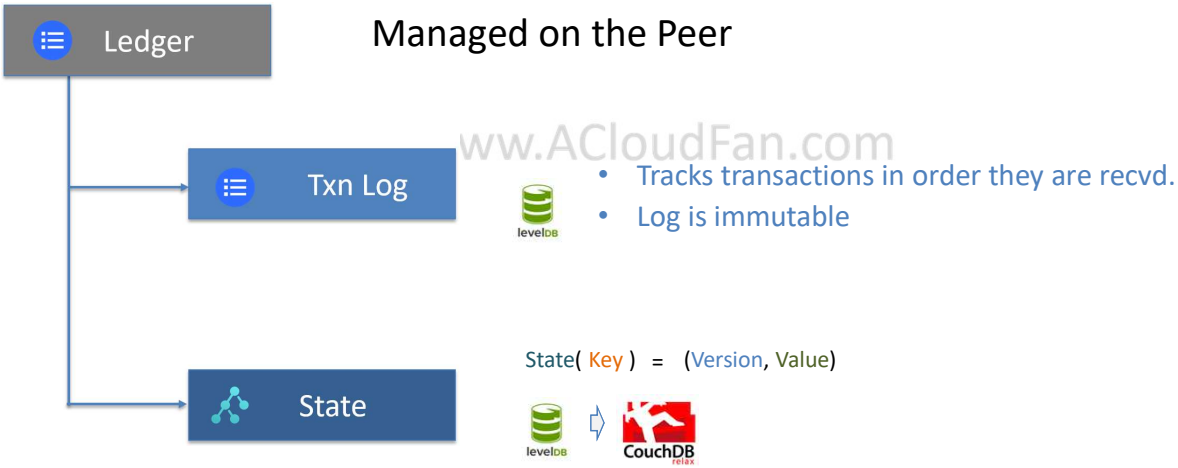
Simple query of values for keys

Execute complex queries




Chain code
Manages
State
Node
Txn Log
Transactions
Channels
Membership Service Providers | CA

# Summary


Ledger
Managed on the Peer
Txn Log
• Tracks transactions in order they are recvd.
• Log is immutable
State
State( Key ) = (Version, Value)
levelDB
CouchDB

# Concepts & Architecture

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

**Discount Coupon Links to UDEMY courses:**

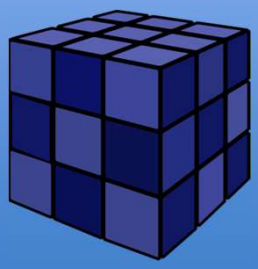
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com





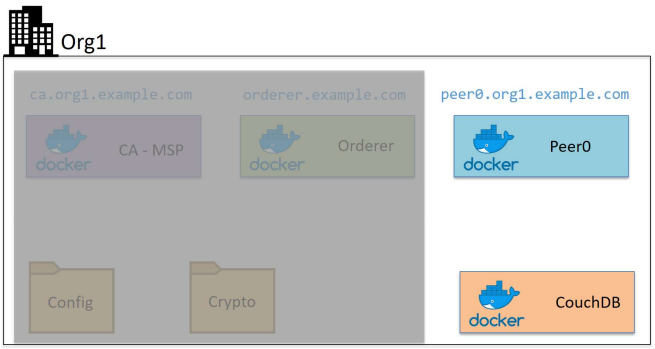
This deck is part of a online course on "Hyperledger Fabric Development with Composer"

# Dev Setup: Ledger

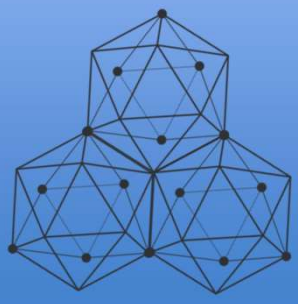
raj@acloudfan.com

@acloudfan

http://ACloudFan.com



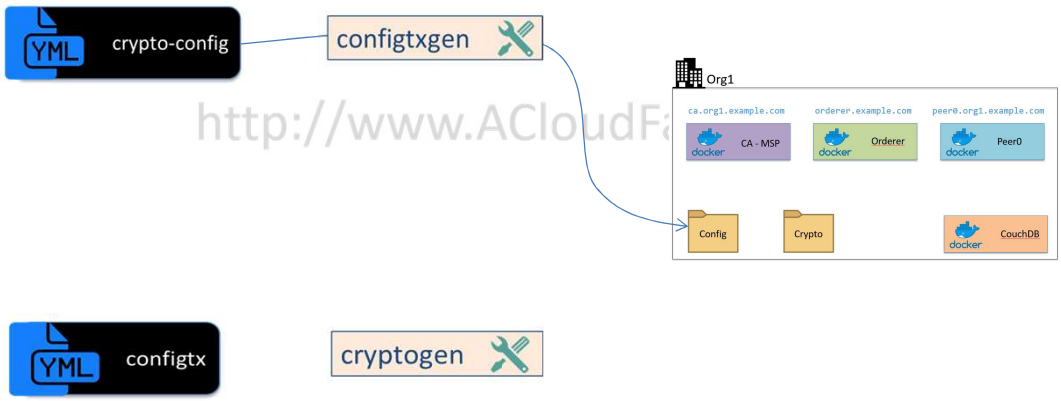


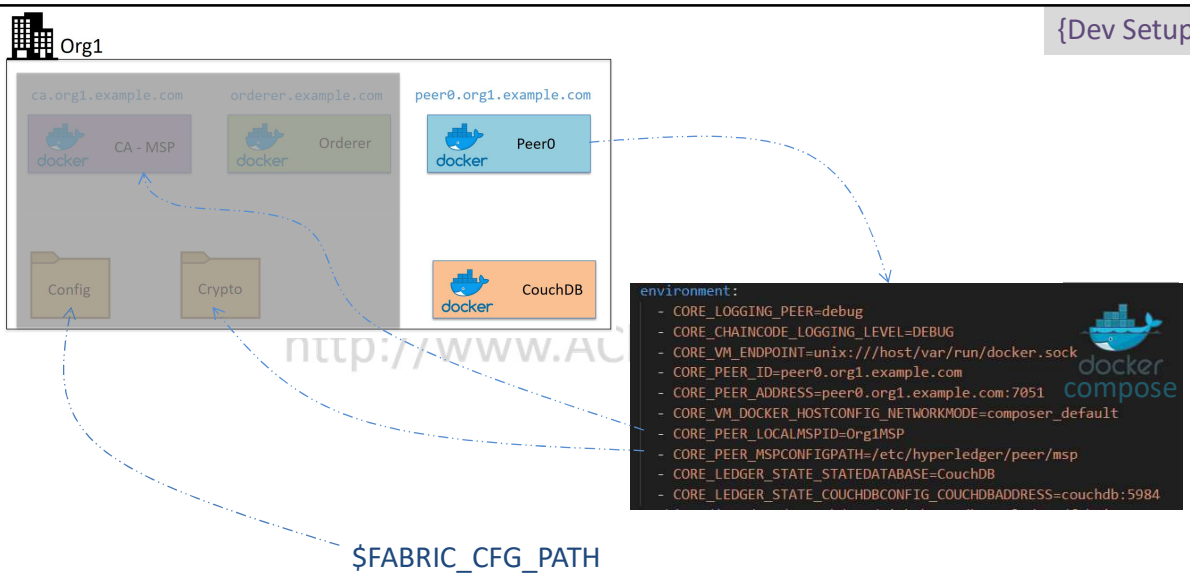



**PS:**

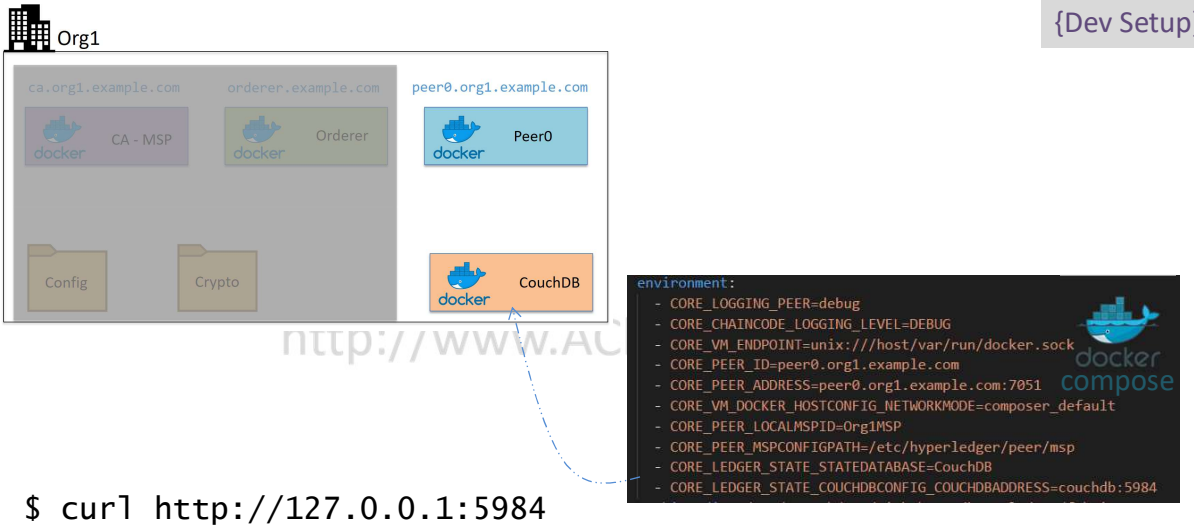
- Infrastructure setup details will not be covered in this course
- Students are encouraged to explore on their own

- Setup configured by way of 2 YAML files



{Dev Setup}




```
$ curl http://127.0.0.1:5984

$ curl http://127.0.0.1:5984/db/_all_dbs
```

# Chaincode

raj@acloudfan.com

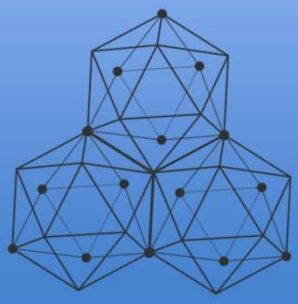
@acloudfan

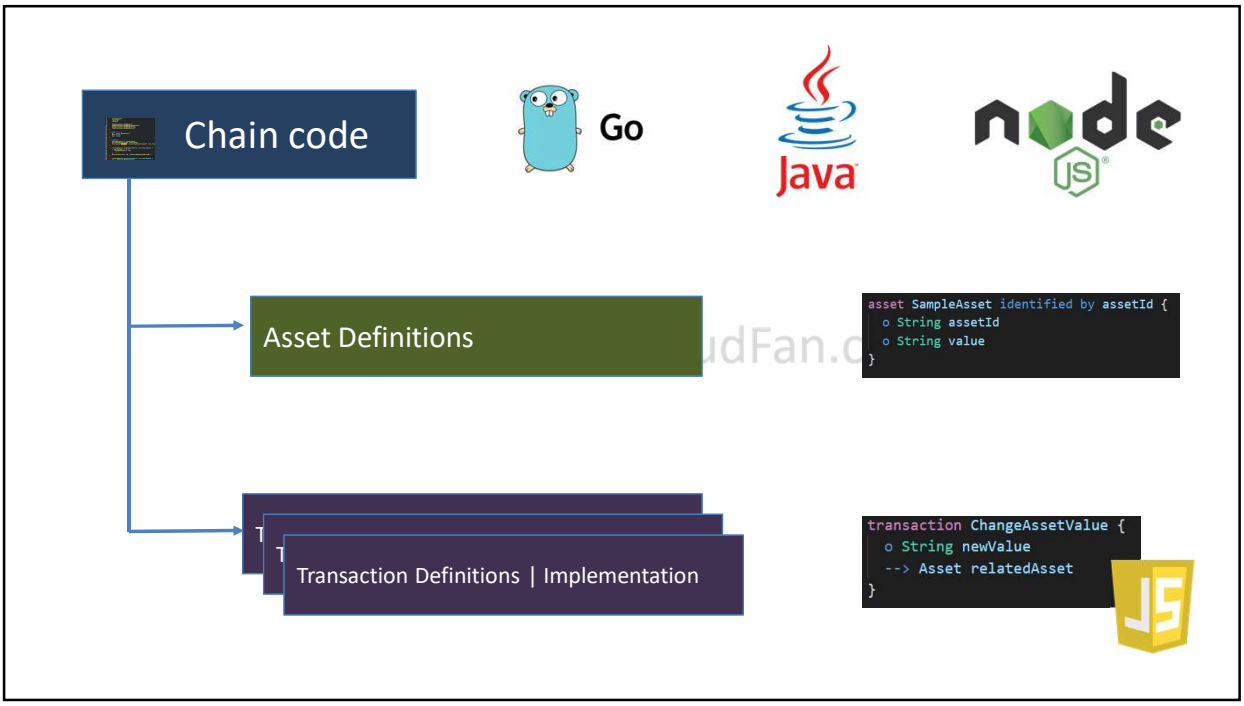
http://ACloudFan.com

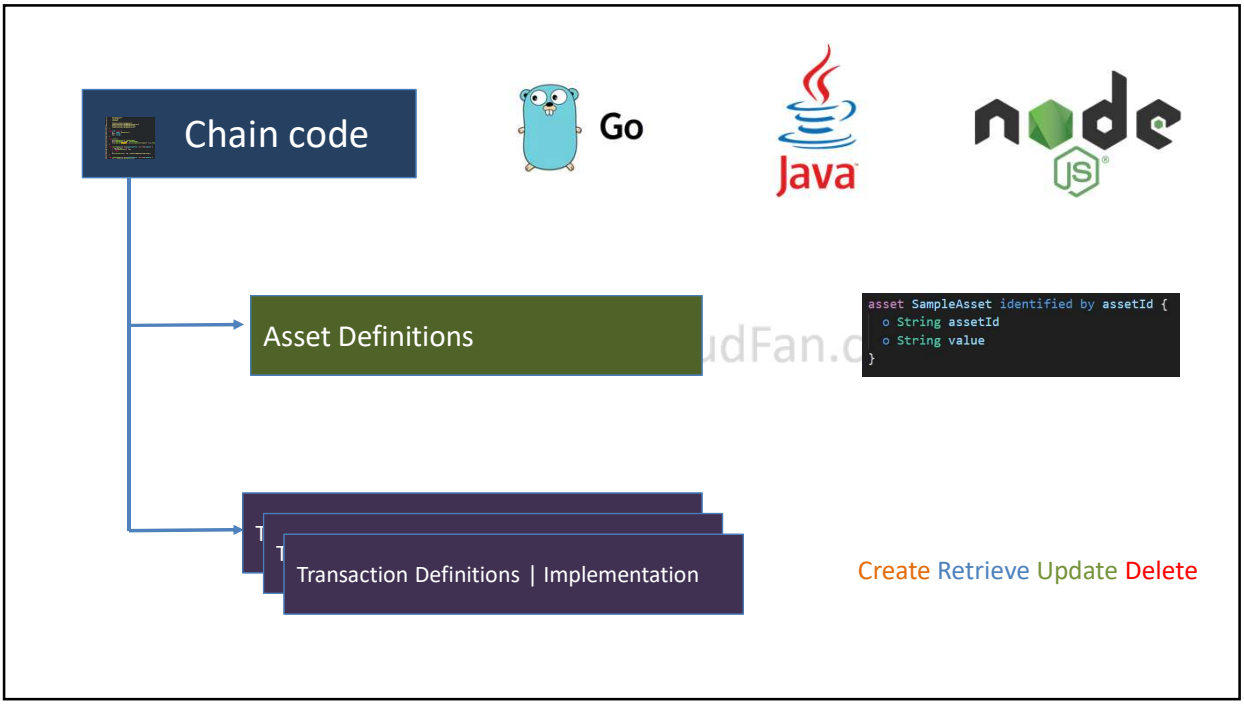
**Learning Objectives:**



- Structure
- Development workflow
- Execution runtime




Chain code
Go
Java
Asset Definitions
asset SampleAsset identified by assetId {
o String assetId
o String value
}
Transaction Definitions | Implementation
transaction ChangeAssetValue {
o String newValue
--> Asset relatedAsset
}


Chain code
Go
Java
Asset Definitions
asset SampleAsset identified by assetId {
o String assetId
o String value
}
Transaction Definitions | Implementation
Create Retrieve Update Delete

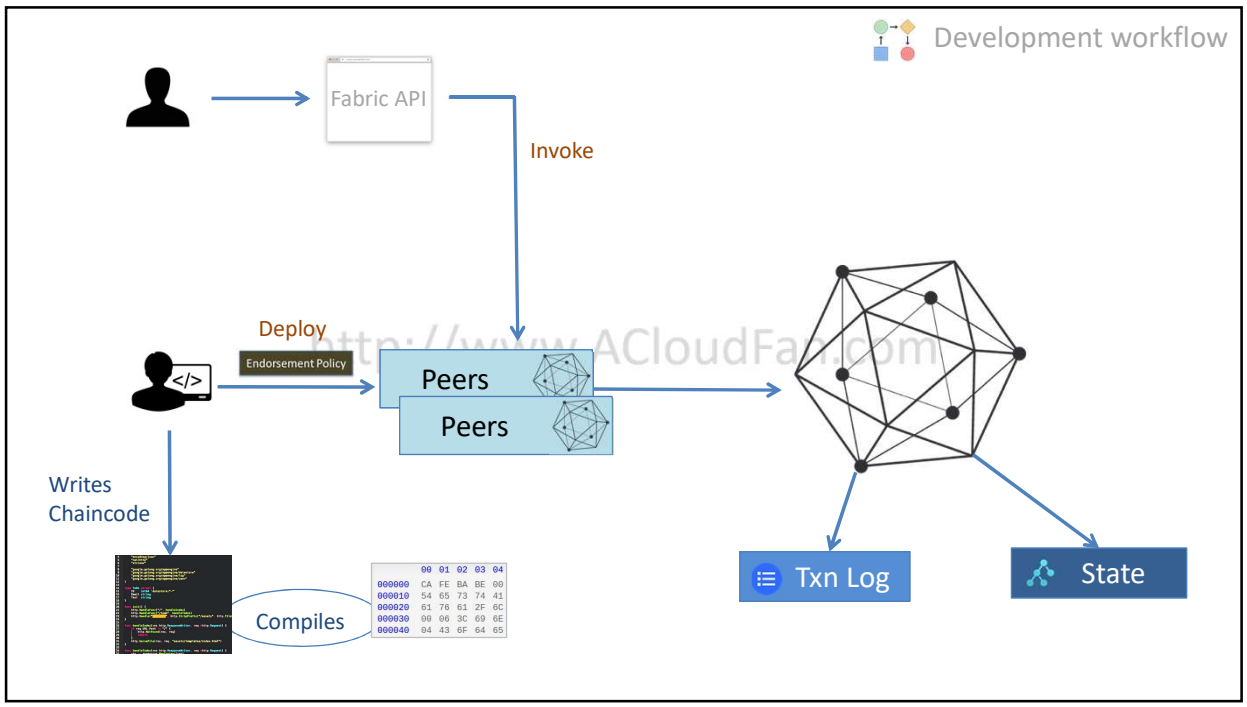
## Chain code

- May specify an Endorsement Policy



Peers

- Instruction on how to validate the transaction



Chain code

This course covers Network Application

Development using Composer



Composer

## Deploy Transaction

- Installs the chaincode on the blockchain
  - Special case of Invoke transaction on a System chaincode

Transaction Payload = Chaincode



  - Recorded in the ledger
  - Governed by System chaincode's *Endorsement Policy*

## Invoke Transaction

- Invokes previously deployed chaincode
  - Specific function is executed

```
stub.GetState(keyName)
```

```
stub.PutState(keyName, value)
```

```
stub.DelState(keyName)
```

  - May lead to state change
  - State changes Recorded in the ledger

## Execution Runtime

- Each Chaincode instance in its own container
- Executed in a Docker container




# Summary

Go

Java

node js

| Asset Definitions | Transaction Definitions \| Implementation |
| --- | --- |
| • Digital representation | • Code for managing asset intsance |



- Deployed to independent

docker

# Peer Nodes

raj@acloudfan.com

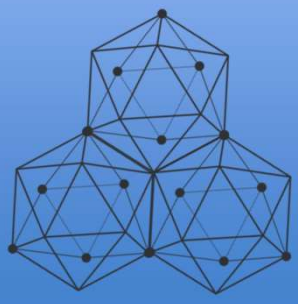
@acloudfan

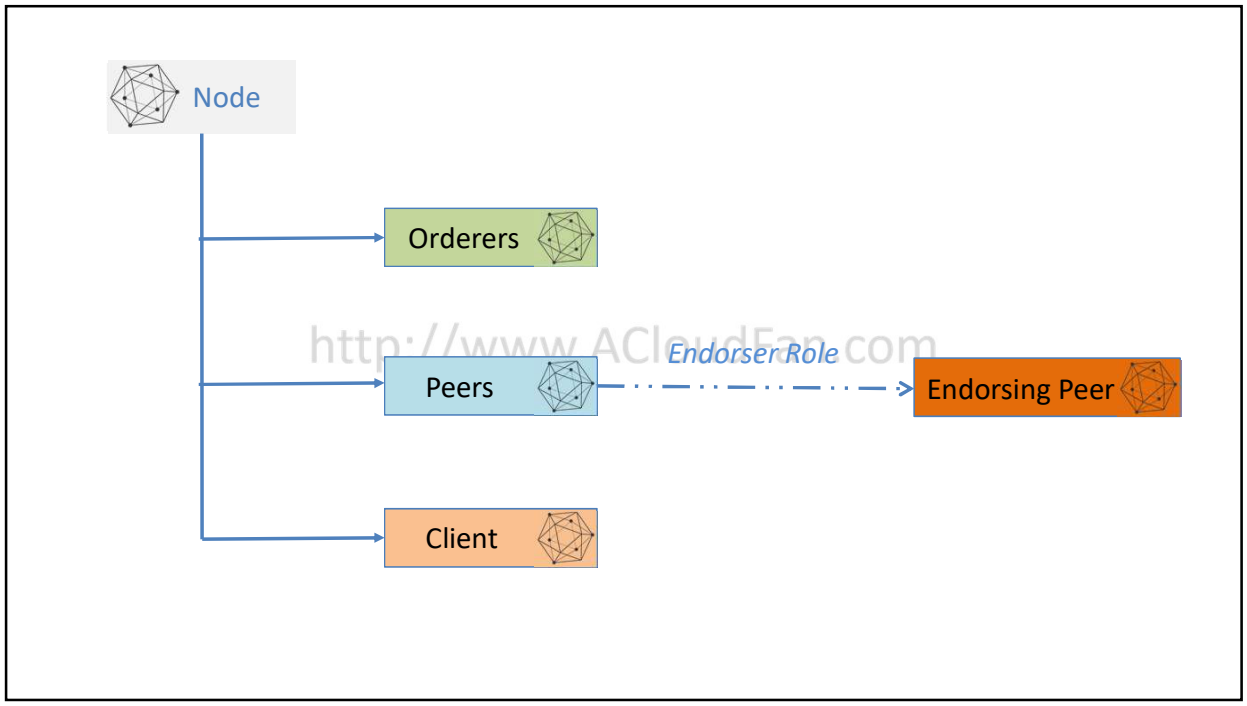
http://ACloudFan.com

**Learning Objectives:**



- Peers
  - Anchor peers
  - Endorsing peers



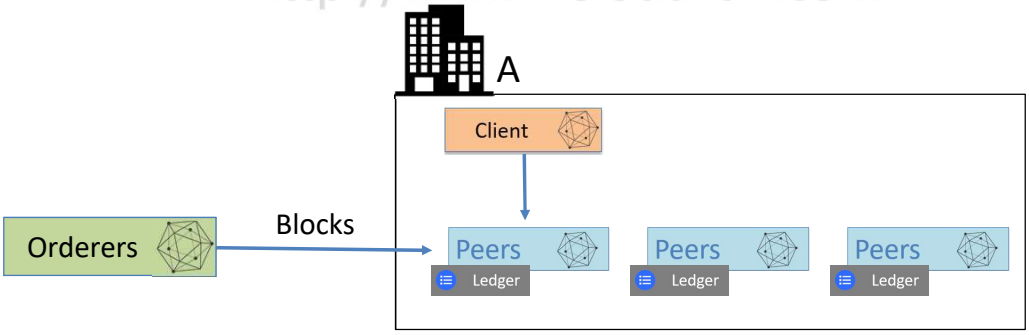





## Peers

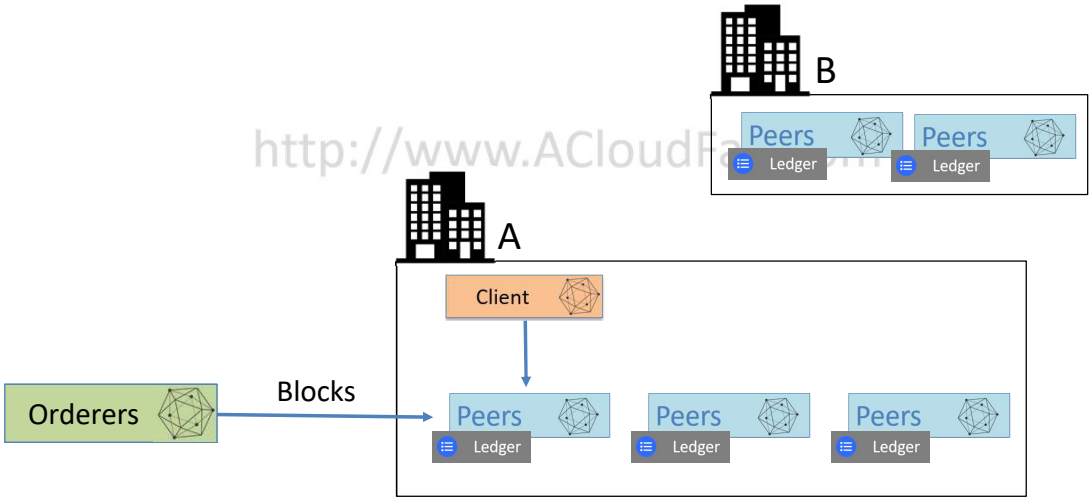
- Receives requests from client
- Local Ledger Synchronization





## Peers

- Highly Scalable Architecture
  - No centralized effort needed

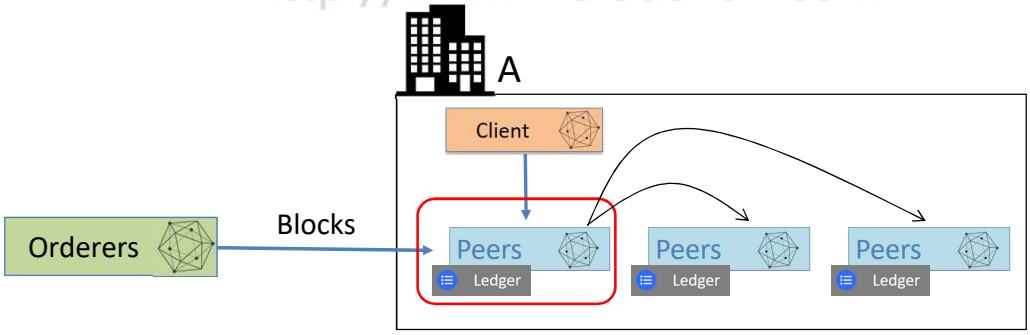





## Peers

- **Anchor Peers** receive the blocks
  - Anchor Peers update other peers in the org
  - Setup at the channel level
  - Discoverable





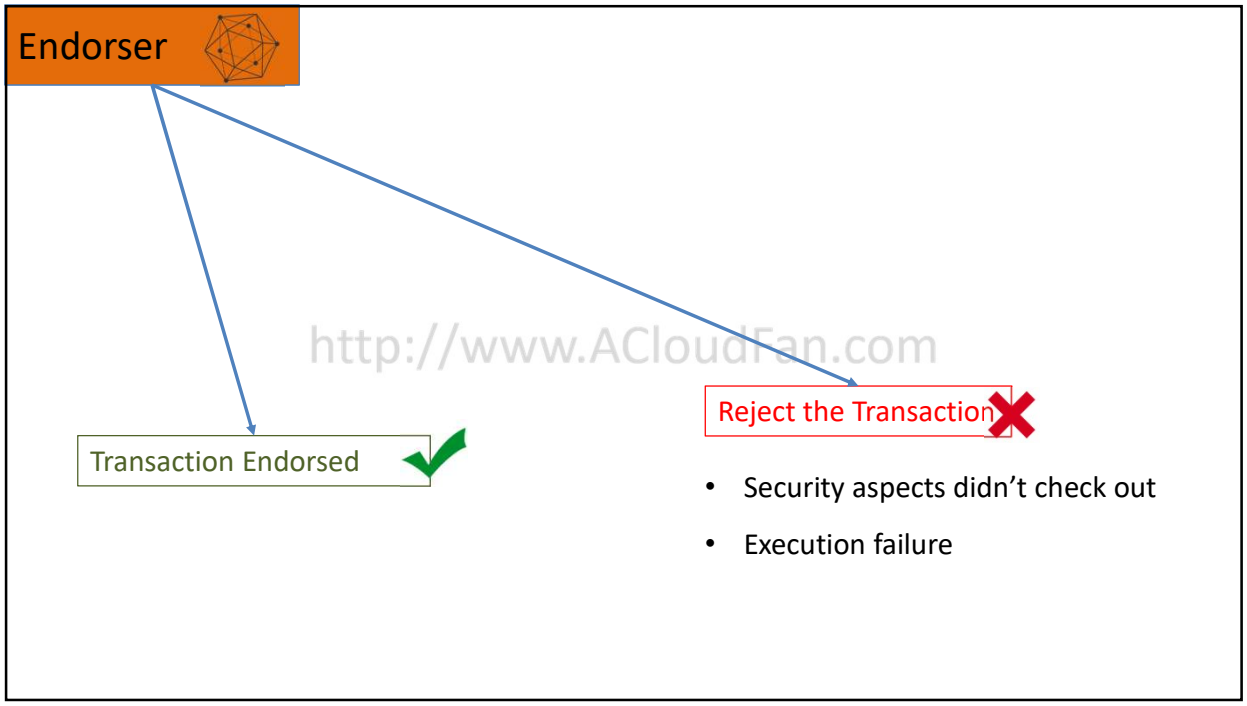
## Endorser

- Peer marked as the endorser a.k.a. endorsing peer
  - Validates the transaction e.g., Certificate checks
  - Simulates the chaincode
    - **Executes** the code
    - But does **NOT** save the state to the Ledger



## Endorser

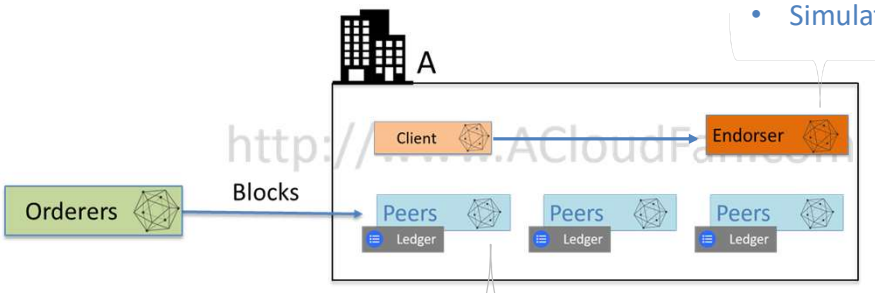
- Primary objective = Protect the network
  - Intentional attack on the network
  - Misbehaving or misconfigured nodes on the network
- Improve scalability as only endorsers need to execute the code
  - NO need for all nodes to execute the chaincode






# Summary

- Protects the network
- Rejects | Accepts
- Simulates the transaction




Anchor Peer

- Receives blocks
- Provide blocks to other peers in the organization

# Client Nodes

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

**Learning Objectives:**



- Client
- Endorsement policy

## Client

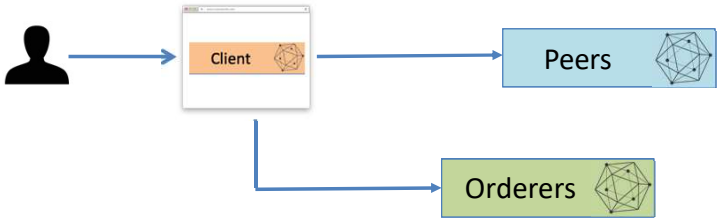
- Client node acts on behalf of the end user
  - a.k.a *Submitting-Client*
- Creates Txn Request and send Endorser(s)



## Client

- Connects to a Orderer for broadcasting validated transaction






## Client

Which peer should be used as an Endorser?

Depends on



Chain code

Defines

Endorsement Policy

1. Which peers to use as endorsers?
2. Criteria for valid transaction?

## Endorsement Policy

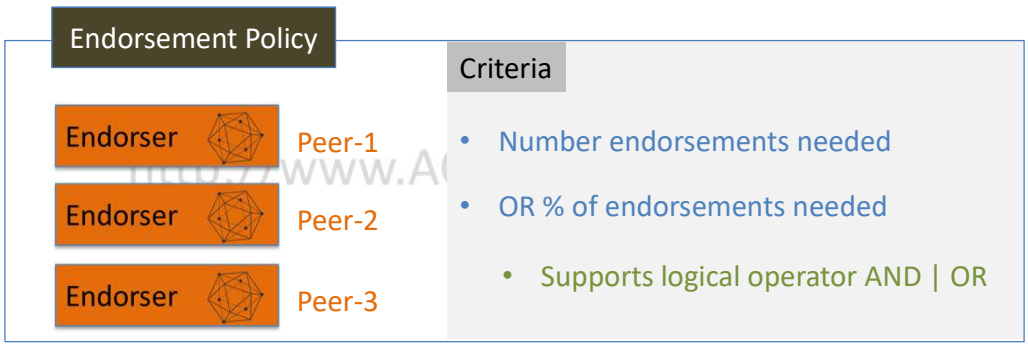
- Association of Endorsement Policy is Optional
- Specified at the time of deployment of chaincode



- Default policy:
  - Any 1 Endorsing peer from default MSP/Organization

## Chain code

Chaincode defines the endorsement policy




```
OR('Org1.member', AND('Org2.member', 'Org3.member'))
```

## Peers

- Uses policy to validate the transaction

1. Are all endorsement valid?
2. Check criteria: Are there enough endorsements?
3. Are the endorsement coming from right sources?

Uses



**Endorsement Policy**

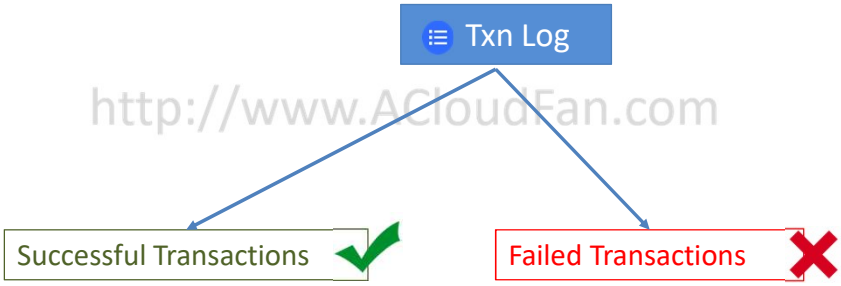
- Endorser — Peer-1
- Endorser — Peer-2
- Endorser — Peer-3

**Criteria**

- Number endorsements needed
- OR % of endorsements needed
  - Supports logical operator AND | OR

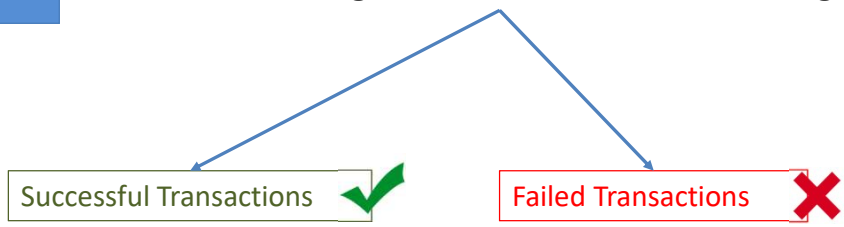
## Peers

Adds the transaction to the ledger



## Txn Log

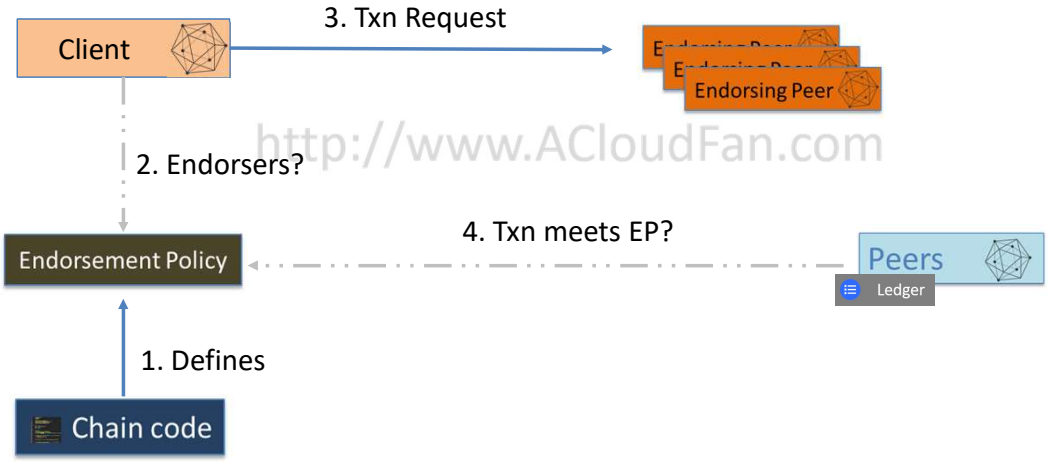
- A chronological record of all state changes






- Transaction history may be used for re-creating the state
  - State data maintained in network may be thought of as optional
- Peers in the network maintain local copy of the ledger

# Summary



# Orderer Nodes

raj@acloudfan.com

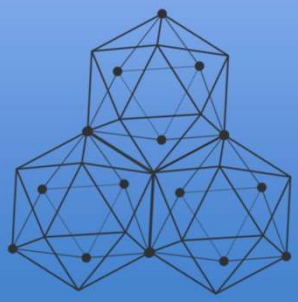
@acloudfan

http://ACloudFan.com

**Learning Objectives:**



- Functions
- Implementation



## Orderer 

Communication channel for fabric

- Also referred to as Ordering Service
- Responsible for consistent ledger state across the network
  - Consensus mechanism
  - Ensures order of transactions
- Creates the blocks & gurantees atomic delivery

## Orderer

Implemented with Message Oriented Middleware

**SOLO**

Single node = Good for development | dev mode

Single point of failure



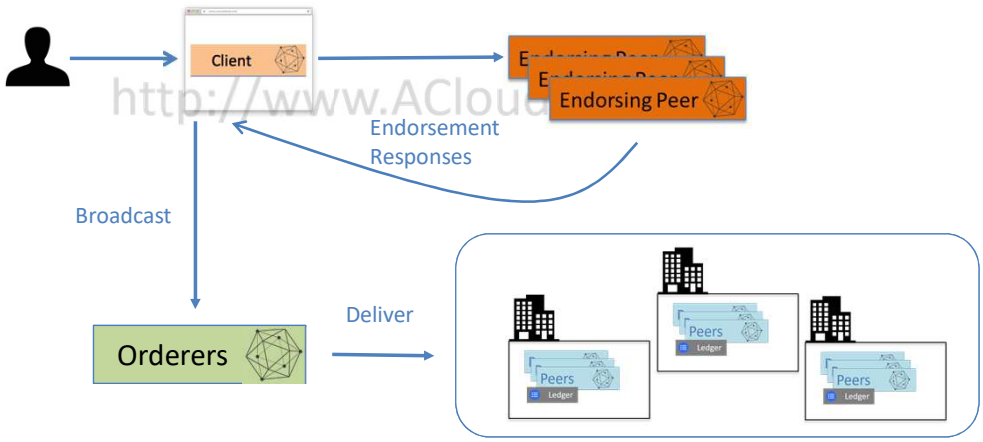
**kafka**

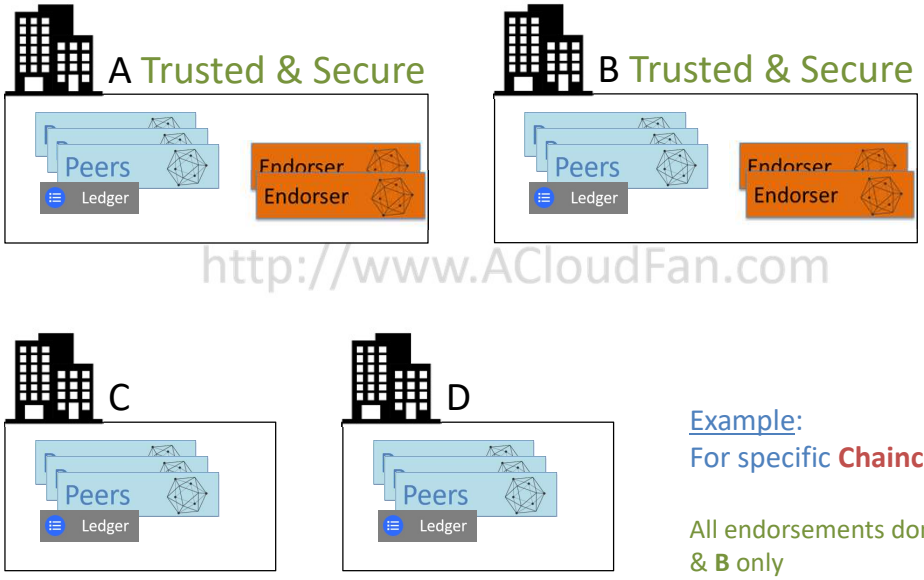
Clustering for high throughput, scalability & fault tolerance

Supports multiple channels | Asynchronous

## Client

- Client broadcasts the endorsed transaction using Orderer








Example:
For specific **ChaincodeID**:

All endorsements done by members **A** & **B** only

# Summary

Orderer

Communication

Order of transactions

Consensus

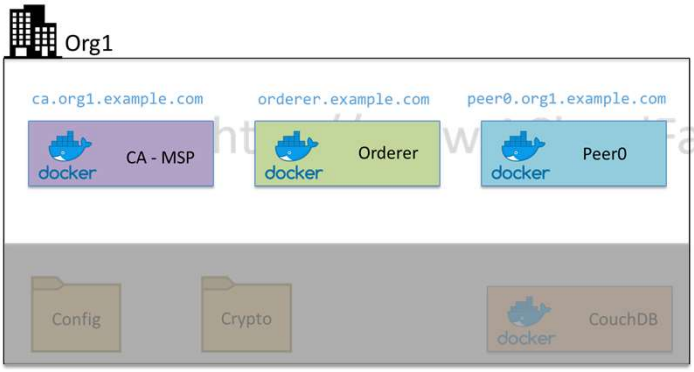


- Implemented with Messaging System

SOLO

kafka

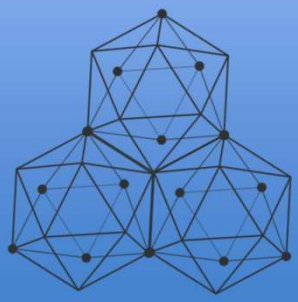
# Dev Setup: Ledger



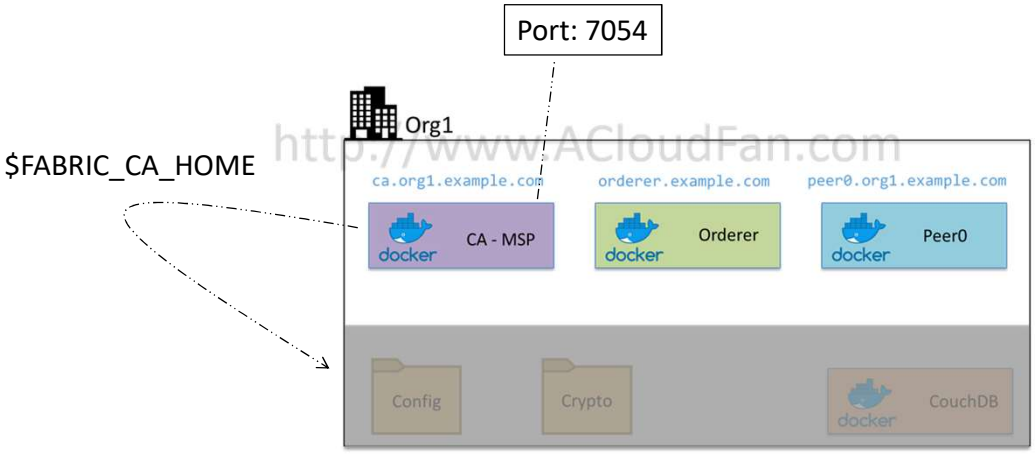

raj@acloudfan.com

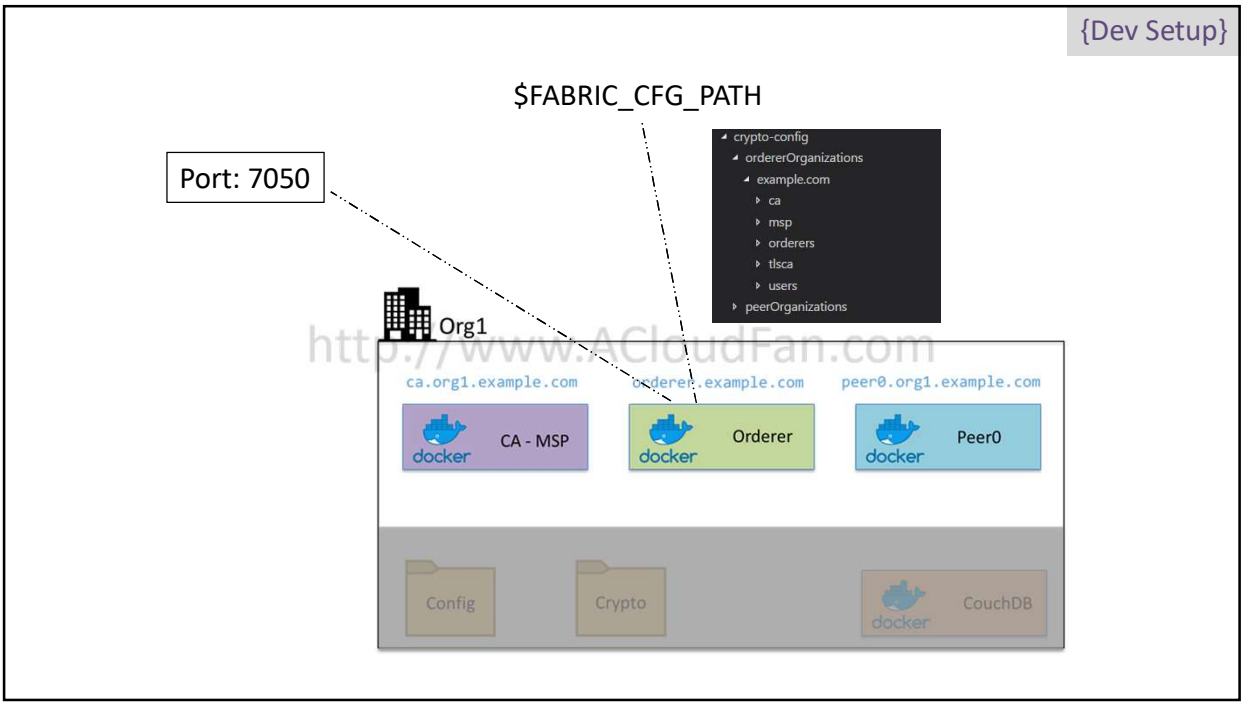
@acloudfan

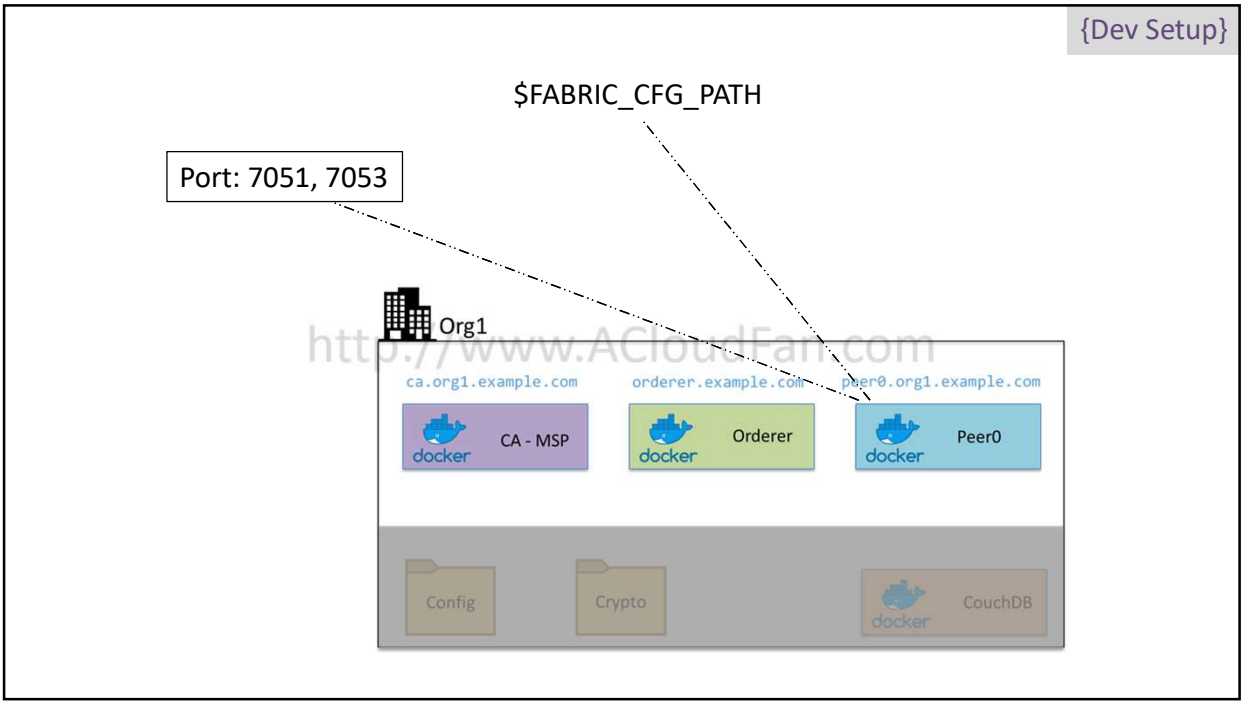
http://ACloudFan.com



{Dev Setup}




{Dev Setup}
$FABRIC_CFG_PATH
Port: 7050
crypto-config
ordererOrganizations
example.com
ca
msp
orderers
tlsca
users
peerOrganizations
Org1
ca.org1.example.com
orderer.example.com
peer0.org1.example.com
CA - MSP
Orderer
Peer0
Config
Crypto
CouchDB


{Dev Setup}
$FABRIC_CFG_PATH
Port: 7051, 7053
Org1
ca.org1.example.com
orderer.example.com
peer0.org1.example.com
CA - MSP
Orderer
Peer0
Config
Crypto
CouchDB