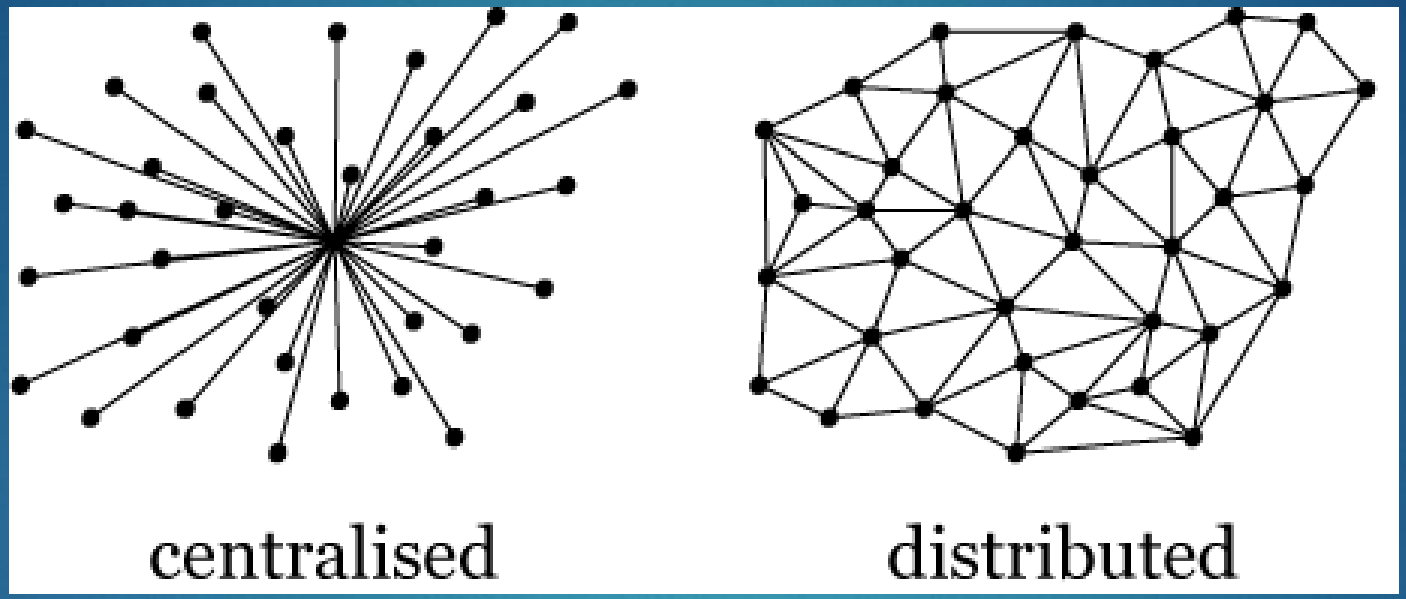
#10 Blockchain

240-311 DISTRIBUTED COMPUTERS AND WEB TECHNOLOGIES (3-0-6)

Outline

- Trust and problem
- Why Blockchains
- How it works
- Blockchain benefits
- Blockchain types
- Hyperledger composer
  - Scenario: carauction

Trust and problem

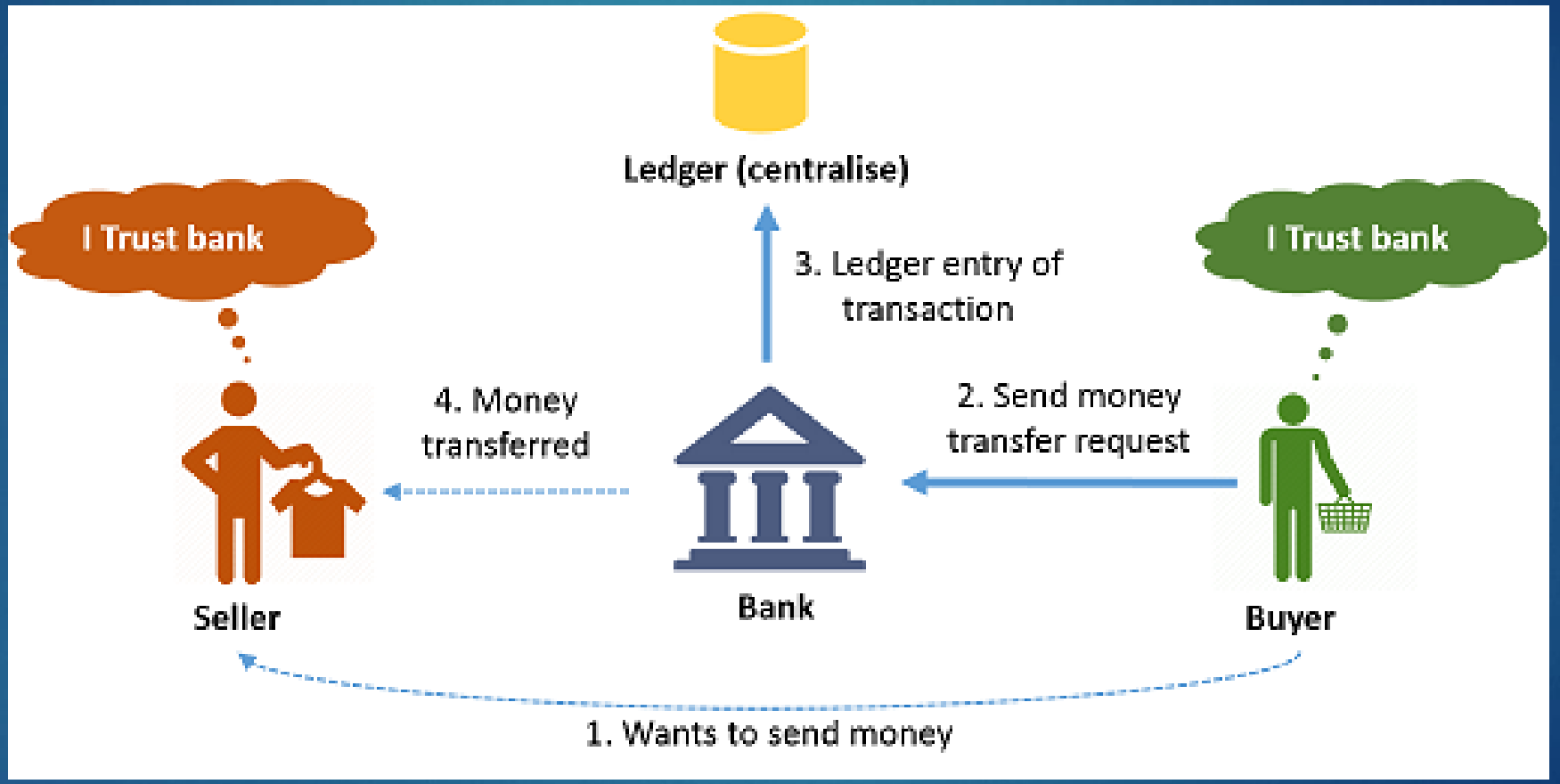


Cons of centralized architecture

- Fault tolerance
- Attack resistance
- Collusion resistance

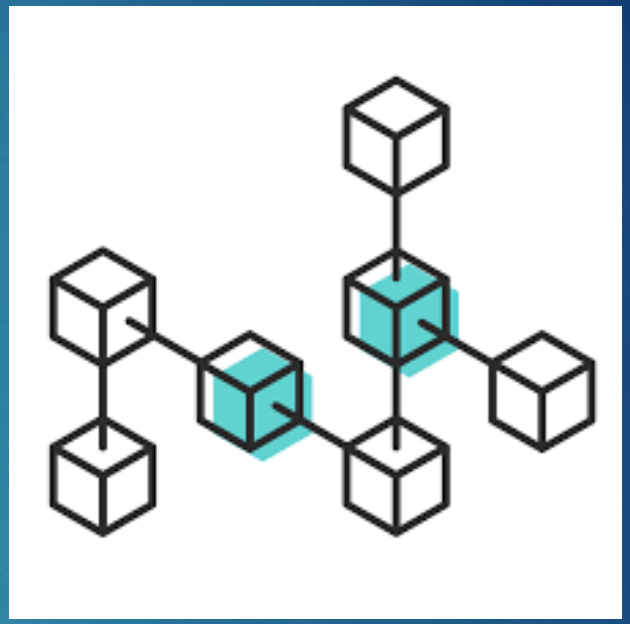
Bitcoin paper: *https://bitcoin.org/bitcoin.pdf*

Why Blockchain

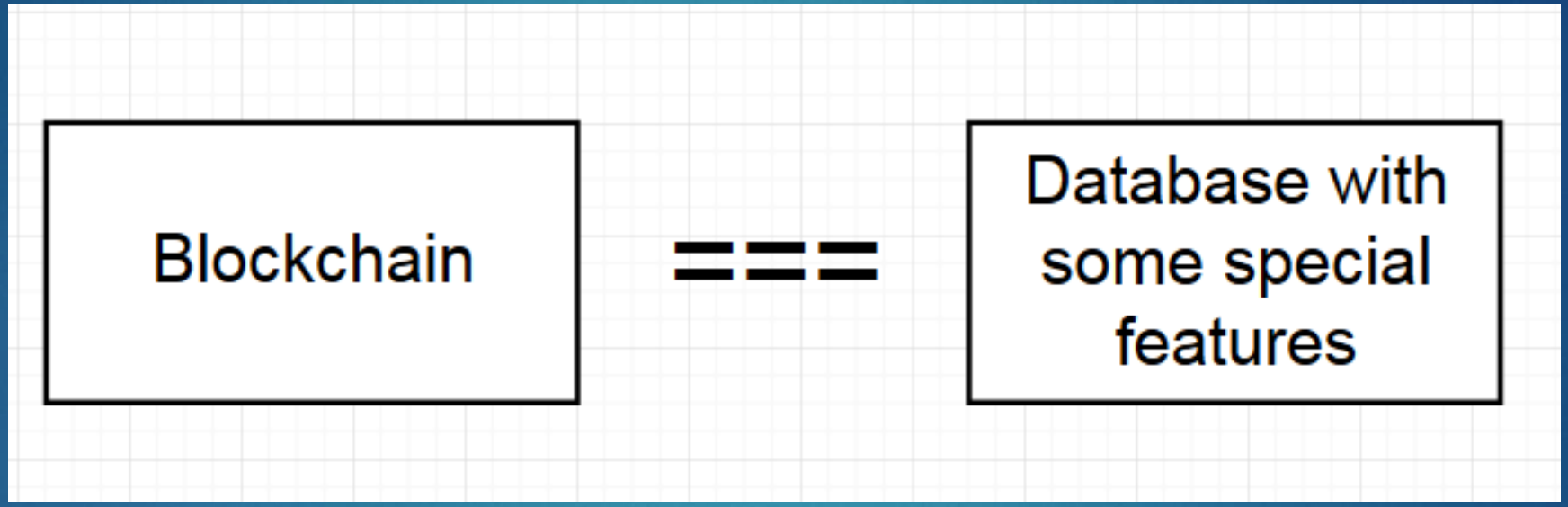


Blockchain

- Blockchain is a growing list of records, called blocks, which are linked using cryptography.
- Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.



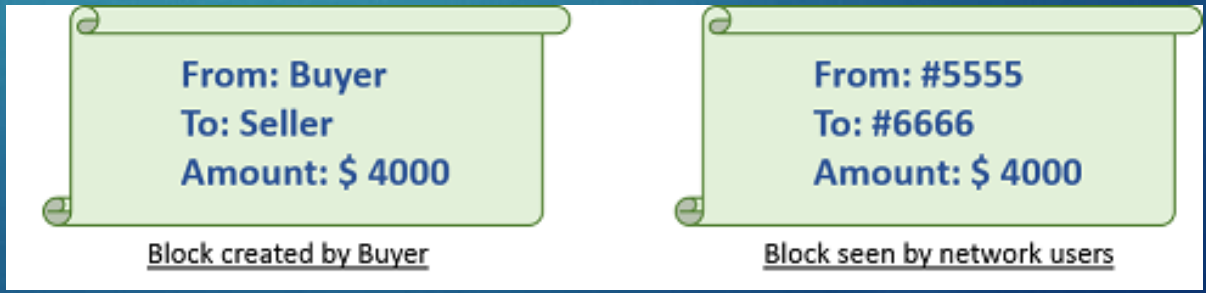
Blockchain



Simple definitions

How Blockchain works?

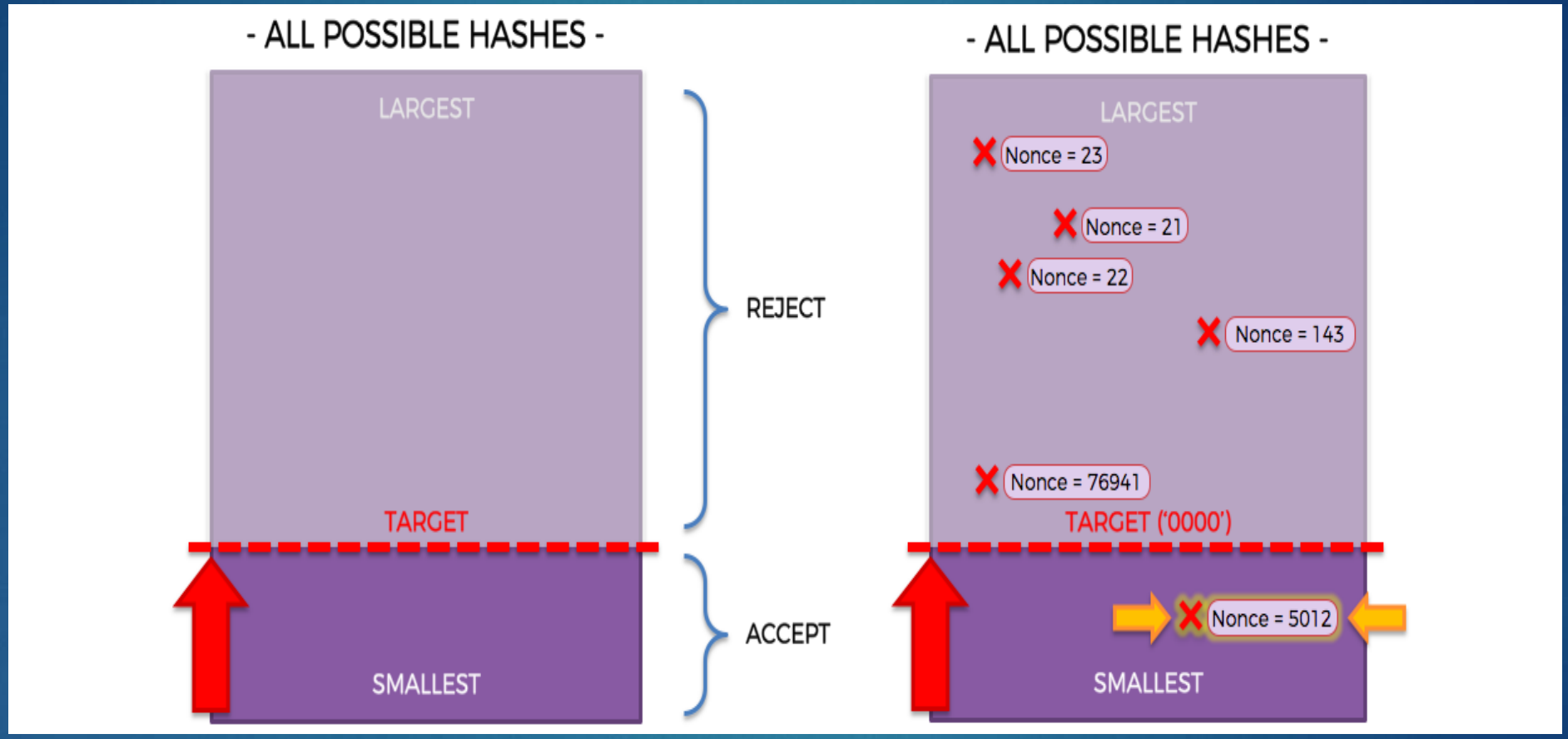
- Remove dependency on third party trust
  - form a group of Members, termed as a NETWORK.
- Terms
  - Members
  - Ledgers
  - Block



Blocks are chained (Linked)

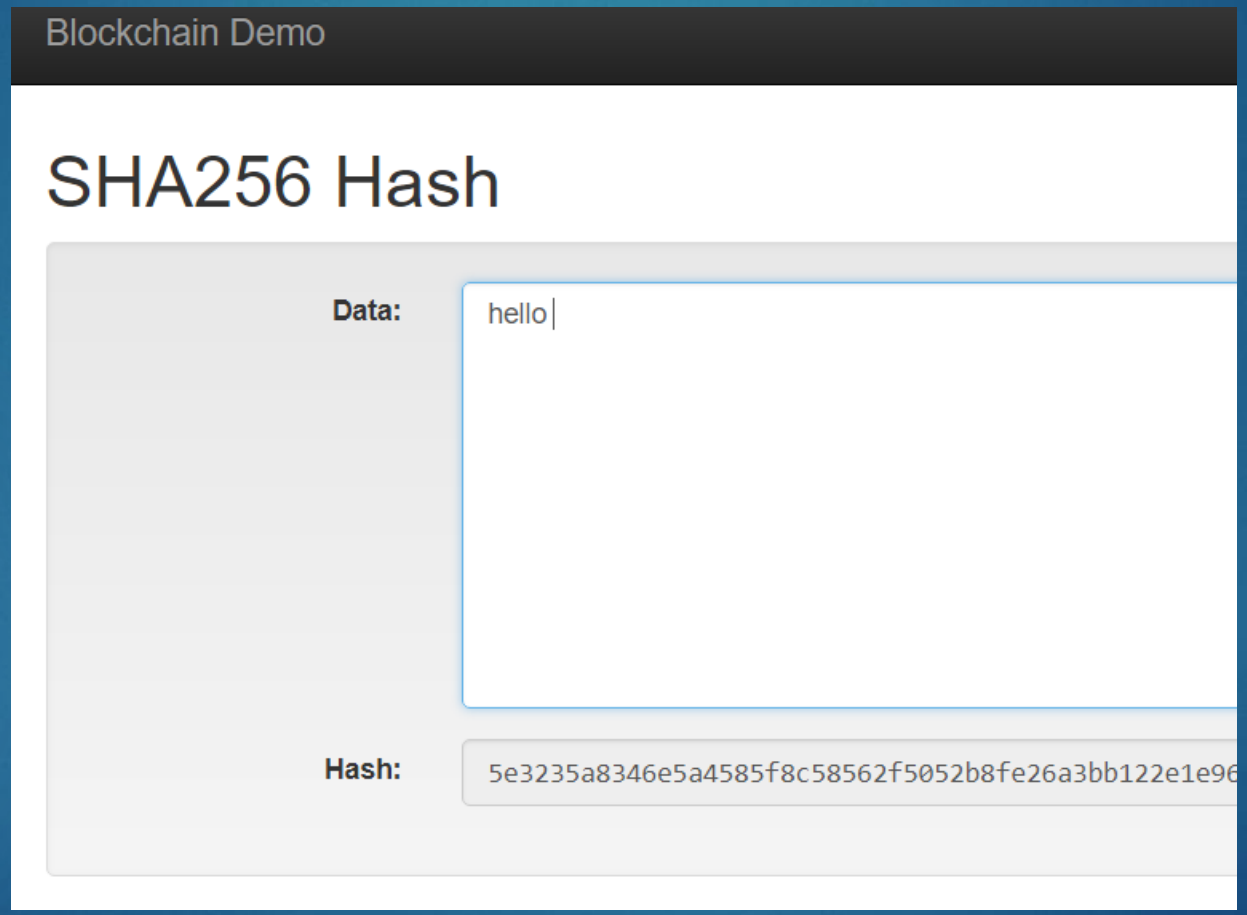
- Hash directly affects the value of the current block's hash.
- if anyone were to tamper with any given block's data, ALL of the following blocks' hashes invalid.

Nonces



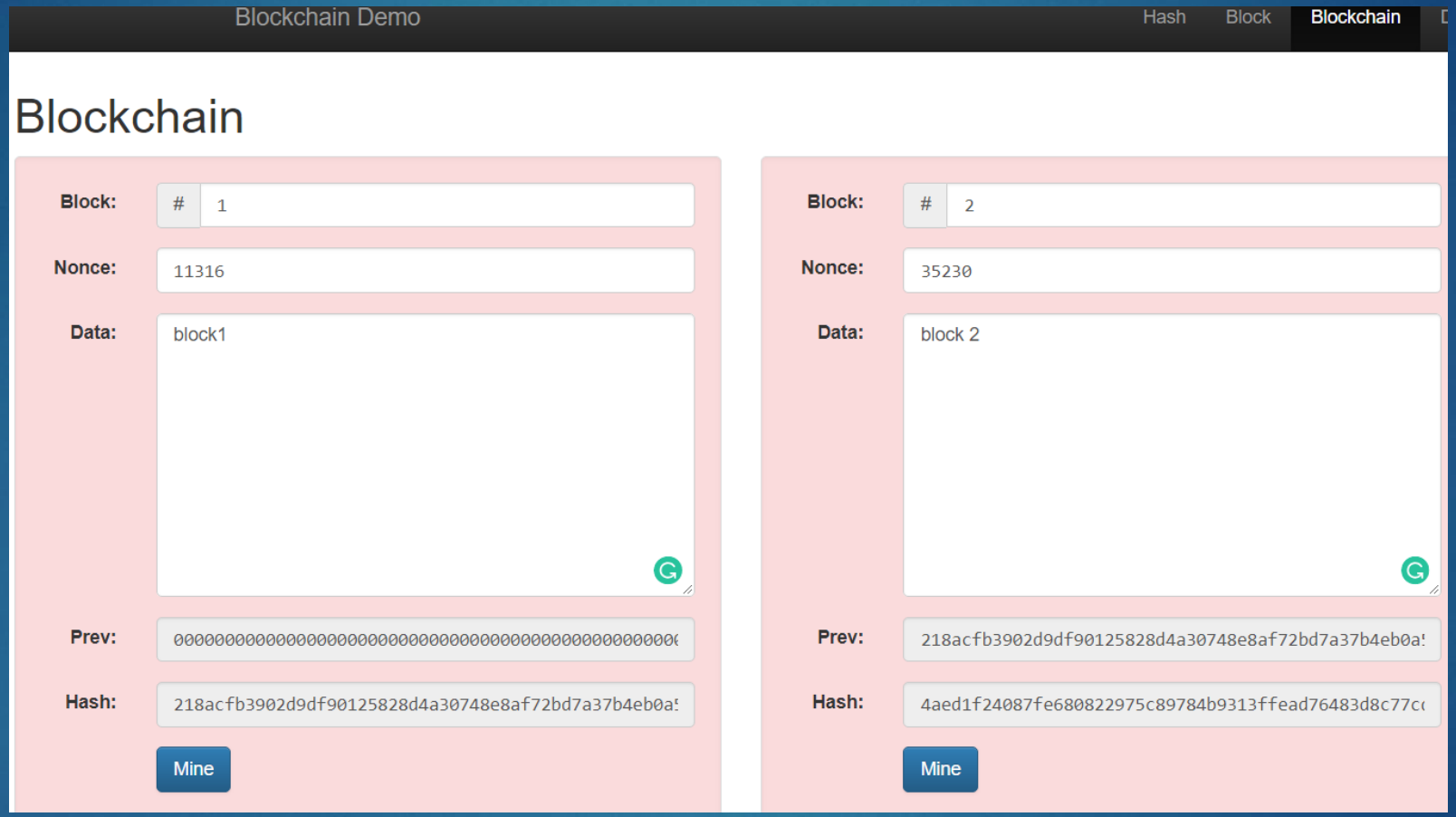
- Miners compete to find a Nonce (also called a Golden Nonce)
- 5012 is the smallest one which is closest to '0000' (4 digits)

Hash demo



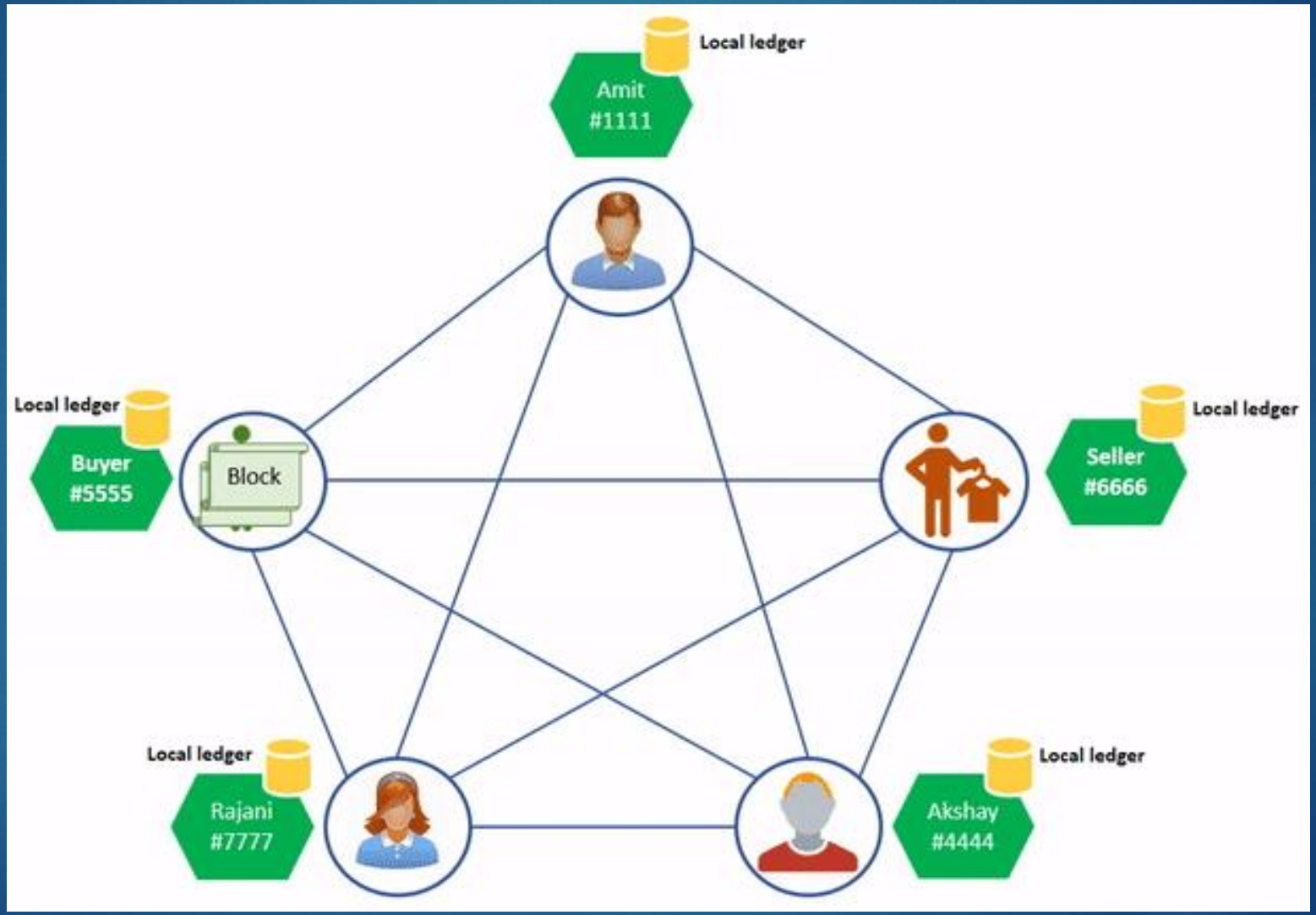
https://anders.com/blockchain/hash.html

Nonce and miner demo



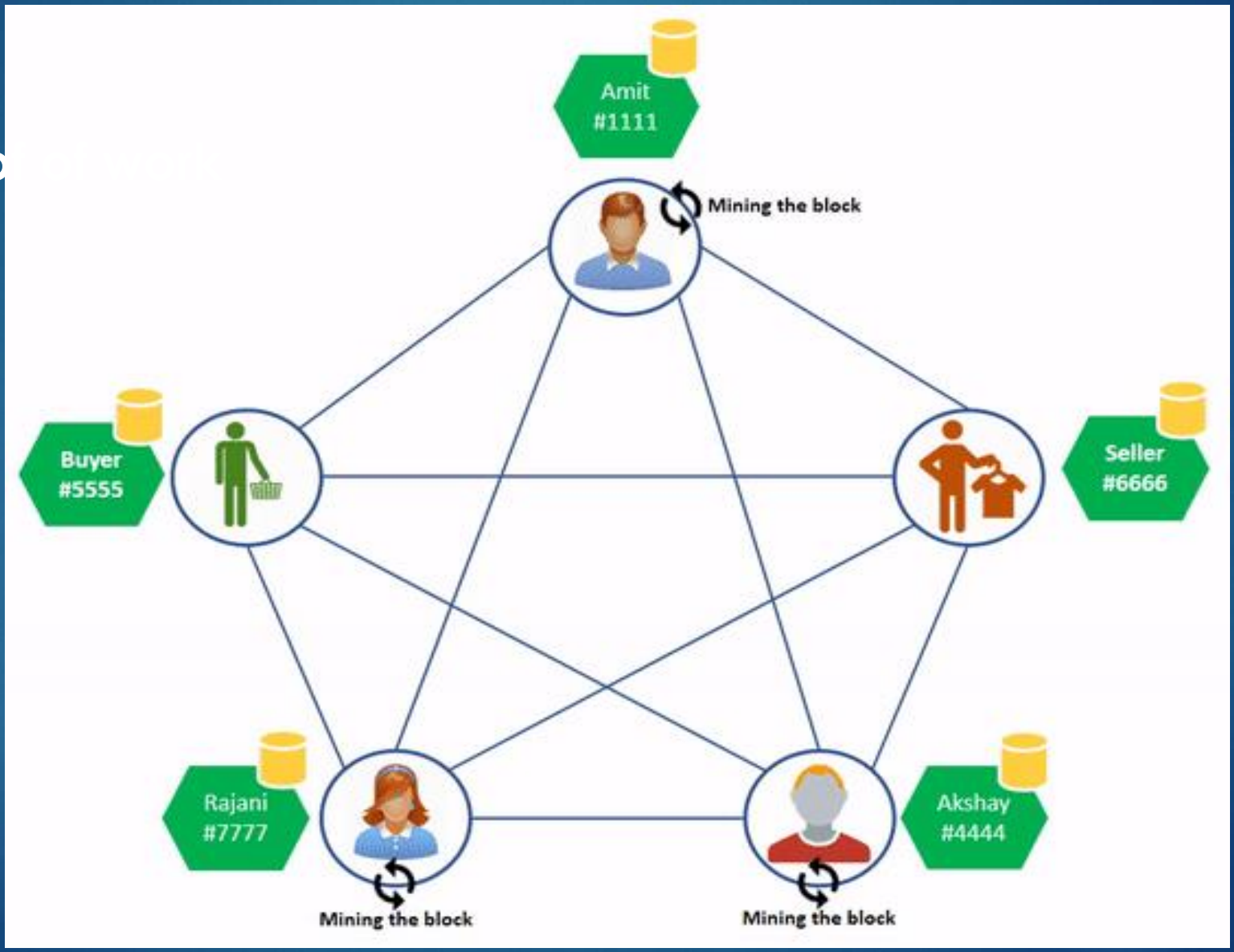
https://anders.com/blockchain/blockchain.html

Distributed Blocks



Mining

Proof of work



Blockchain benefits

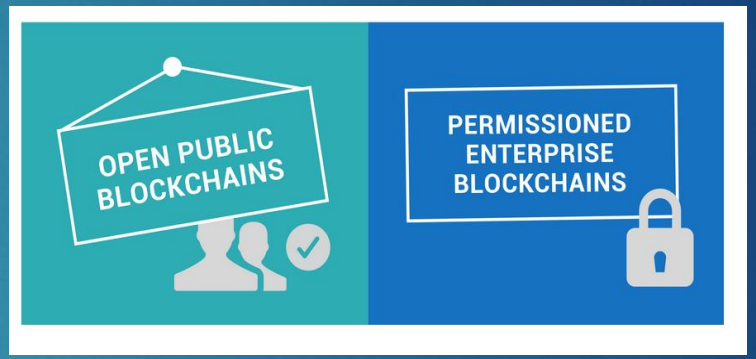
- Traceability
- Enhanced security
  - Update through consensus
  - Immutable (process integrity)
- Efficiency and speed
  - Fast processing with distributed technologies
- Reduced costs

Blockchain scenario examples

- Electronic voting
- Car auction
- Land lord and title deed
  - In the case of untrusted government
    - Blockchain helps to prove that you are the real landlord without a title deed issued by untrusted government
- Room/Hotel business
  - No any centralized agent, room owners and renters are directly connected through smart contracts
- Money changer
  - Less exchange fee, fast (without physical boundary) and secure

Blockchain types

- Public Blockchains
  - All participants are anonymous members
    - Not suitable for business
  - Competition (Proof of Works)
  - Crypto currency & Reward



- Private Blockchains (or Permissioned Blockchains)
  - Permissioned DLT – Distributed Hyperledgers
  - Identity module & Confidential transaction
  - No Cryptocurrency & programmable (automate business logic)
  - No competition (and incentive) since participants are identified
    - Participants will be kicked out if they are cheating

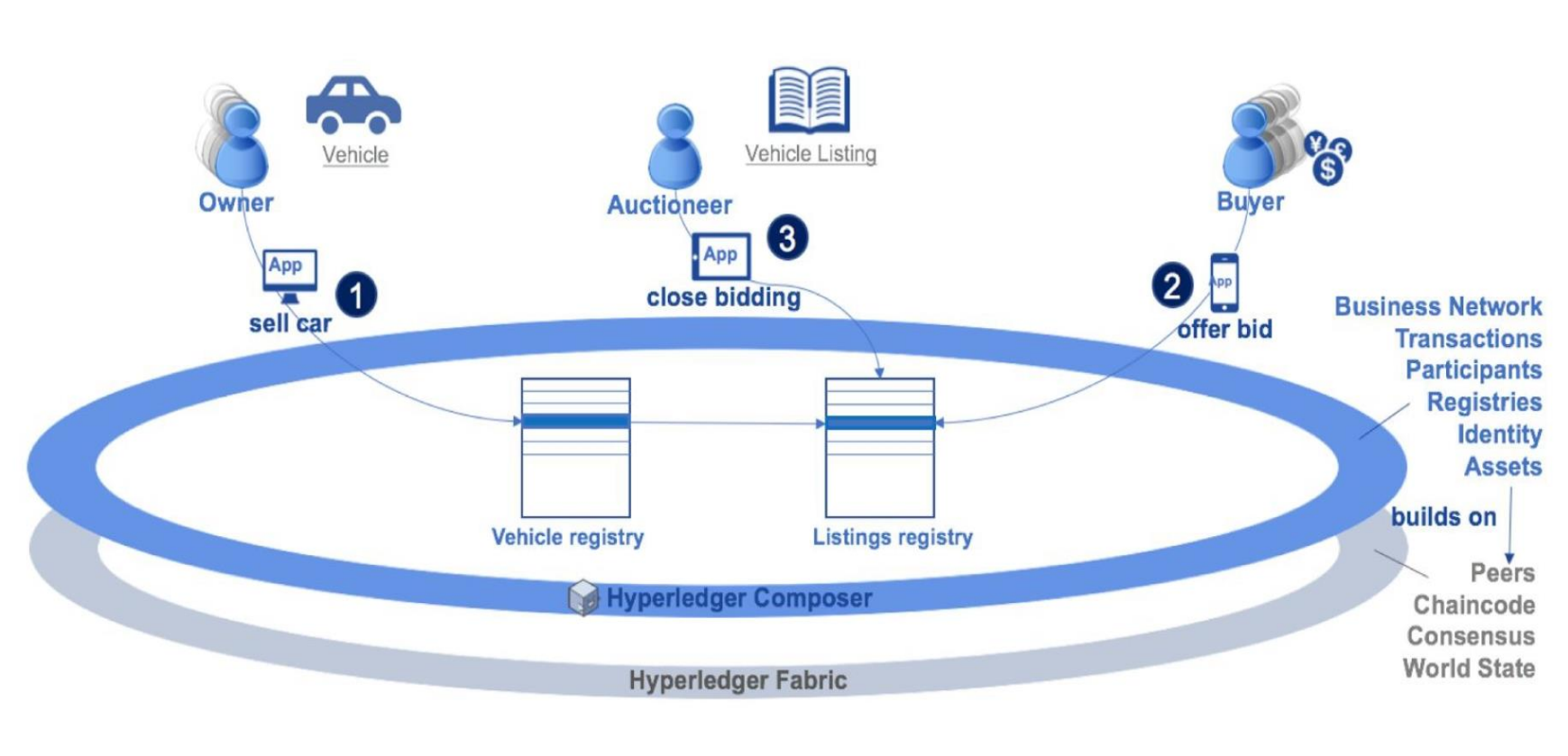
Private Blockchains

- Enterprise blockchains developer
- Tools for full stack blockchains:
  - Hyperledger composer
    - Focus on business network:
      - participants, identity, assets, transaction
    - Blockchain applications (DApp)
  - Hyperledger fabric
    - Focus on peers, chaincode, consensus
    - Backend blockchains

Hyperledger Composer

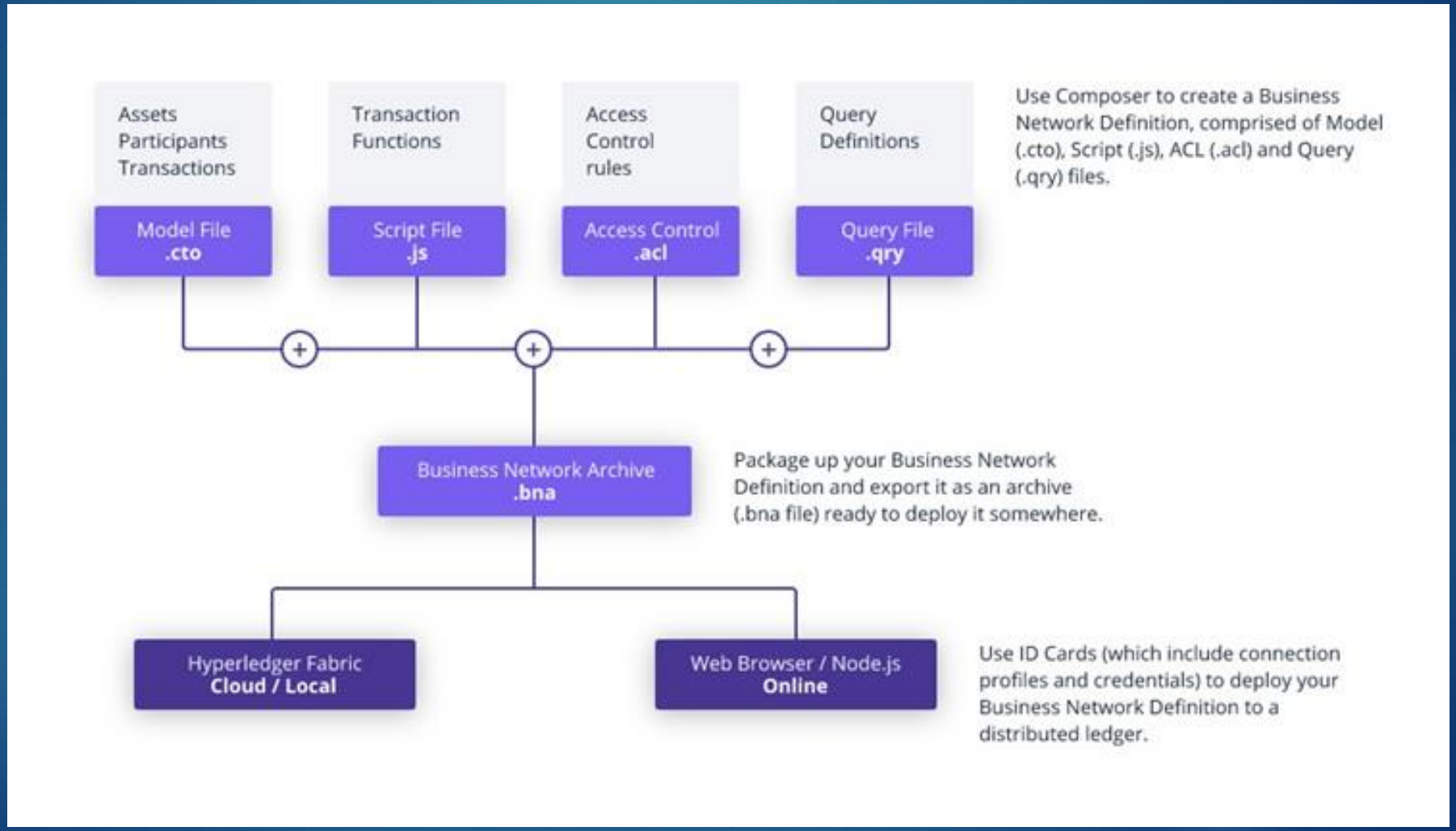
- Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier.
- It simplifies application development on top of the Hyperledger Fabric blockchain infrastructure
  - which allows components, such as consensus and membership services, to be plug-and-play.
- Model a business network and integrate existing systems and data with the blockchain applications.

Hyperledger Composer & Fabric



Hyperledger Composer

Composer abstracts the Blockchain complexities



Hyperledger composer experiment

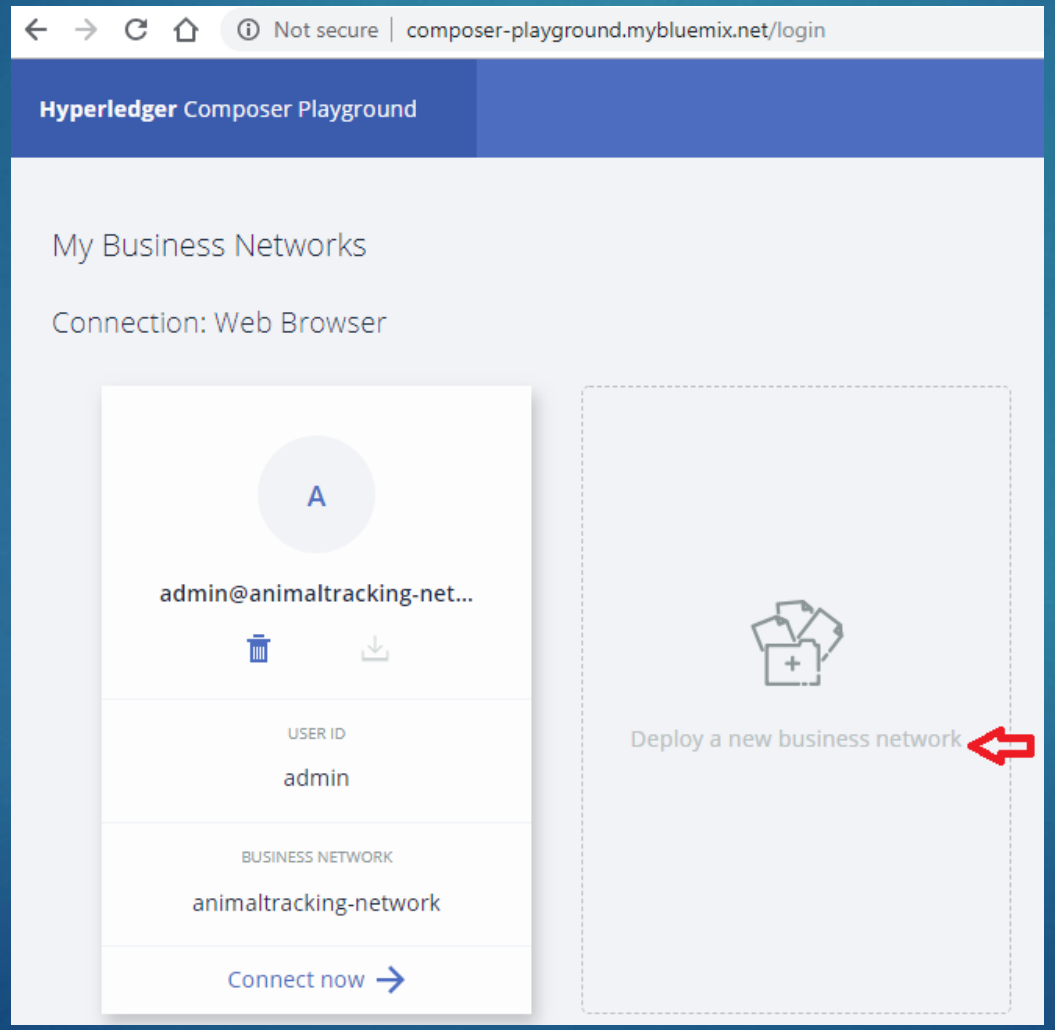
CARAUCTION

Business network: Car Auction

- The model implements:
  - 3 member participants:
    - Paul: Owner,
    - Warodom and Kevin: Buyer (bidding)
  - Two assets:
    - A vehicle: a car with id (9999)
    - A vehicle listing: declare for bidding offer (carListing)
  - Two transactions:
    - Making an offer (bid) on a car and closing a bid on an auction.

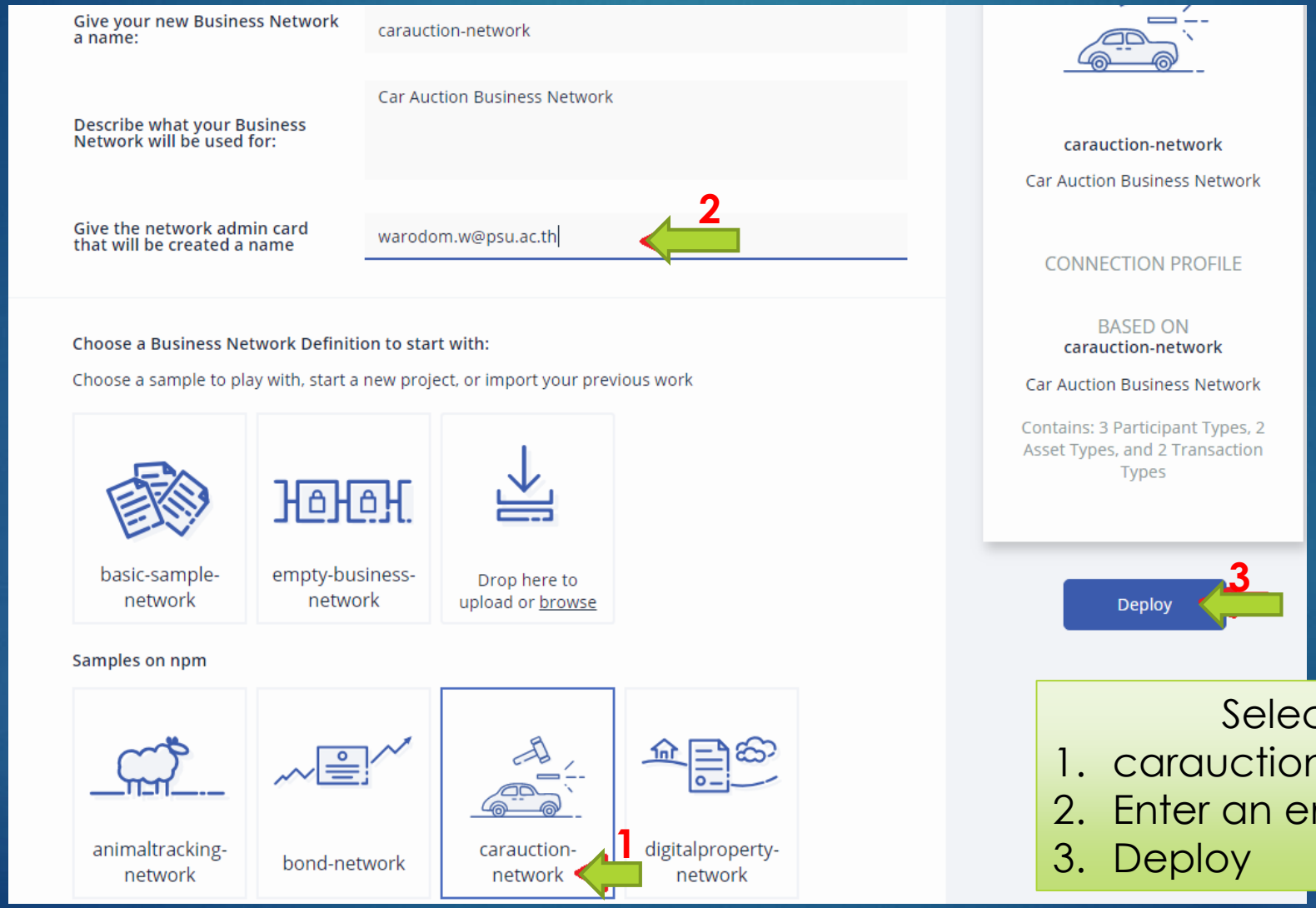
Experiment

- http://composer-playground.mybluemix.net/



Select deploy a new business network

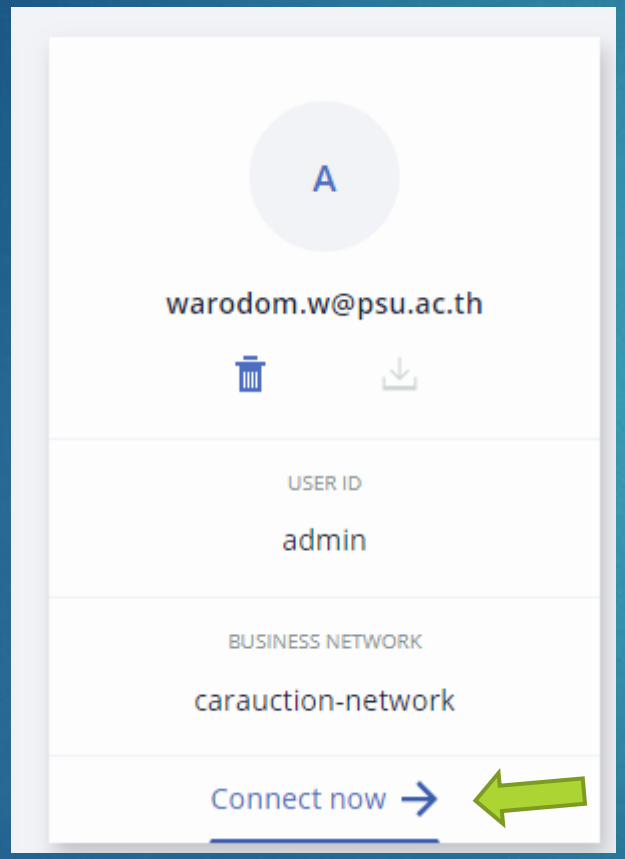
Experiment



Select

1. carauction-network
2. Enter an email
3. Deploy

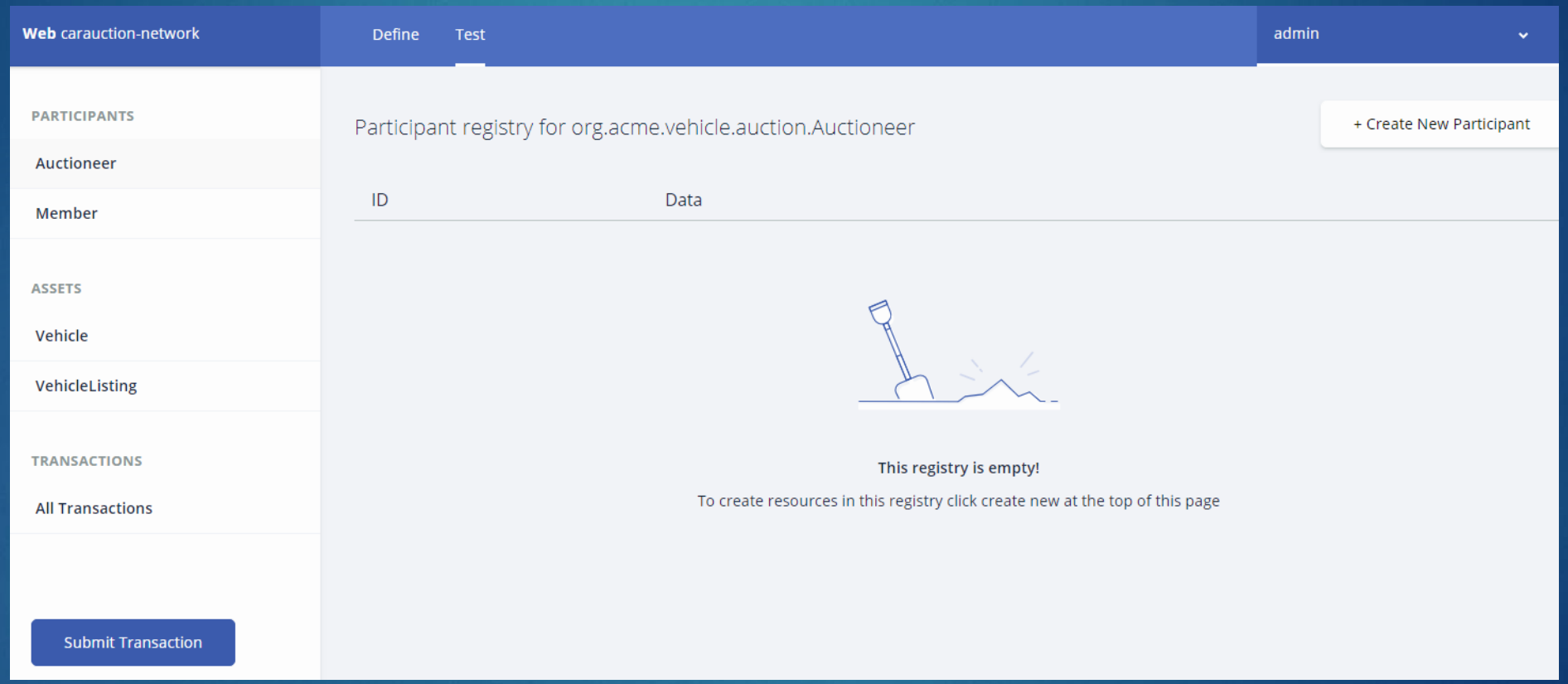
Connect to carauction



Connect now

- Hyperledger fabric for blockchain network is provided by the web site.
- We focus on Business network (caracution) as DApp only.
- Scenario:
  - 1 member who wants to sale a car
  - 2 members who wants to buy a car from this auction system

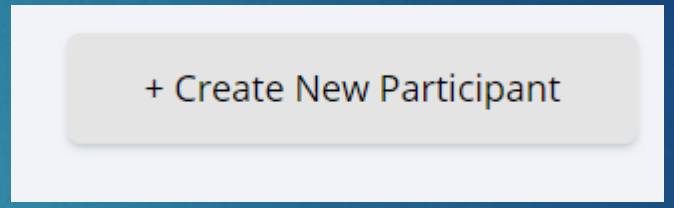
Test carauction network



Ready to implement business network

Create participants

- 3 member participants
  - Paul Gilbert as a car owner
  - Warodom Werapun as a buyer
  - Kevin Durant as another buyer
- No Auctione



In registry: **org.acme.vehicle.auction.Member**

JSON Data Preview

```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 100,
  "email": "paul@psu.ac.th",
  "firstName": "Paul",
  "lastName": "Gilbert"
}
```

Create 2 more participants


```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 90000,
  "email": "warodom@psu.ac.th",
  "firstName": "Warodom",
  "lastName": "Werapun"
}
```

```
{
  "$class": "org.acme.vehicle.auction.Member",
  "balance": 10000,
  "email": "kevin@psu.ac.th",
  "firstName": "Kevin",
  "lastName": "Durant"
}
```

All member participants

ID	Data
kevin@psu.ac.th	{ "$class": "org.acme.vehicle.auction.Member", "balance": 10000, "email": "kevin@psu.ac.th", "firstName": "Kevin", "lastName": "Durant" } Collapse
paul@psu.ac.th	{ "$class": "org.acme.vehicle.auction.Member", "balance": 100, "email": "paul@psu.ac.th", "firstName": "Paul", "lastName": "Gilbert" Show All
warodom@psu.ac.th	{ "$class": "org.acme.vehicle.auction.Member", "balance": 90000, "email": "warodom@psu.ac.th", "firstName": "Warodom", "lastName": "Werapun" Show All

Create asset and asset listing

In registry: **org.acme.vehicle.auction.Vehicle**

JSON Data Preview

Asset id (car id)

```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "vin": "9999",
  "owner":
"resource:org.acme.vehicle.auction.Member#paul@psu.ac.th"
}
```

Car owner

In registry: **org.acme.vehicle.auction.VehicleListing**

JSON Data Preview

```
{
  "$class": "org.acme.vehicle.auction.VehicleListing",
  "listingId": "carListing",
  "reservePrice": 1000,
  "description": "Paul wants to sale his car",
  "state": "FOR_SALE",
  "vehicle": "resource:org.acme.vehicle.auction.Vehicle#9999"
}
```

Asset and asset listing

car

Data


```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "vin": "9999",
  "owner": "resource:org.acme.vehicle.auction.Member#paul@psu.ac.th"
}
```


carListing

Data


```
{
  "$class": "org.acme.vehicle.auction.VehicleListing",
  "listingId": "carListing",
  "reservePrice": 1000,
  "description": "Paul wants to sell his car",
  "state": "FOR_SALE",
  "vehicle": "resource:org.acme.vehicle.auction.Vehicle#9999"
}
```

Submit transaction

Transaction Type: Offer

JSON Data Preview

```
{
  "$class": "org.acme.vehicle.auction.Offer",
  "bidPrice": 80000,
  "listing":
"resource:org.acme.vehicle.auction.VehicleListing#carListing",
  "member":
"resource:org.acme.vehicle.auction.Member#kevin@psu.ac.th"
}
```

Transaction Type: Offer

JSON Data Preview

```
{
  "$class": "org.acme.vehicle.auction.Offer",
  "bidPrice": 200000,
  "listing":
"resource:org.acme.vehicle.auction.VehicleListing#carListing",
  "member":
"resource:org.acme.vehicle.auction.Member#warodom@psu.ac.th"
}
```

Bidding Offer

- Kevin places 'Bidding offer' at 80,000 baht
- Warodom offer at 200,000 baht
- Paul will close bidding and let see the winner

After place bidding offer

Data


```
{
  "$class": "org.acme.vehicle.auction.VehicleListing",
  "listingId": "carListing",
  "reservePrice": 1000,
  "description": "Paul wants to sale his car",
  "state": "FOR_SALE",
  "offers": [
    {
      "$class": "org.acme.vehicle.auction.Offer",
      "bidPrice": 80000,
      "listing": "resource:org.acme.vehicle.auction.VehicleListing#carListing",
      "member": "resource:org.acme.vehicle.auction.Member#kevin@psu.ac.th",
      "transactionId": "58ff01da-3982-4882-98dd-f6deb1e61d6e",
      "timestamp": "2019-02-02T06:50:24.945Z"
    },
    {
      "$class": "org.acme.vehicle.auction.Offer",
      "bidPrice": 200000,
      "listing": "resource:org.acme.vehicle.auction.VehicleListing#carListing",
      "member": "resource:org.acme.vehicle.auction.Member#warodom@psu.ac.th",
      "transactionId": "b0baf88d-f9b5-42aa-8bd9-0c5677f911f3",
      "timestamp": "2019-02-02T06:57:40.741Z"
    }
  ],
  "vehicle": "resource:org.acme.vehicle.auction.Vehicle#9999"
}
```

Submit transaction

Transaction Type: CloseBidding

JSON Data Preview

```
1  {
2    "$class": "org.acme.vehicle.auction.CloseBidding",
3    "listing":
   "resource:org.acme.vehicle.auction.VehicleListing#carListing"
4  }
```

Data

```
{
  "$class": "org.acme.vehicle.auction.VehicleListing",
  "listingId": "carListing",
  "reservePrice": 1000,
  "description": "Paul wants to sale his car",
  "state": "SOLD",
  "vehicle": "resource:org.acme.vehicle.auction.Vehicle#9999"
}
```

State is changed

Place close bidding


```
async function closeBidding(closeBidding) {  // eslint-disabl
    const listing = closeBidding.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    // by default we mark the listing as RESERVE_NOT_MET
    listing.state = 'RESERVE_NOT_MET';
    let highestOffer = null;
    let buyer = null;
    let seller = null;
    if (listing.offers && listing.offers.length > 0) {
        // sort the bids by bidPrice
        listing.offers.sort(function(a, b) {
            return (b.bidPrice - a.bidPrice);
        });
        highestOffer = listing.offers[0];
        if (highestOffer.bidPrice >= listing.reservePrice) {
            // mark the listing as SOLD
            listing.state = 'SOLD';
            buyer = highestOffer.member;
            seller = listing.vehicle.owner;
            // update the balance of the seller
```

Smart contract in lib/logic.js is called:
async function closeBidding(closeBidding)

Asset owner is changed

carListing is
lly updated
be “SOLD”.

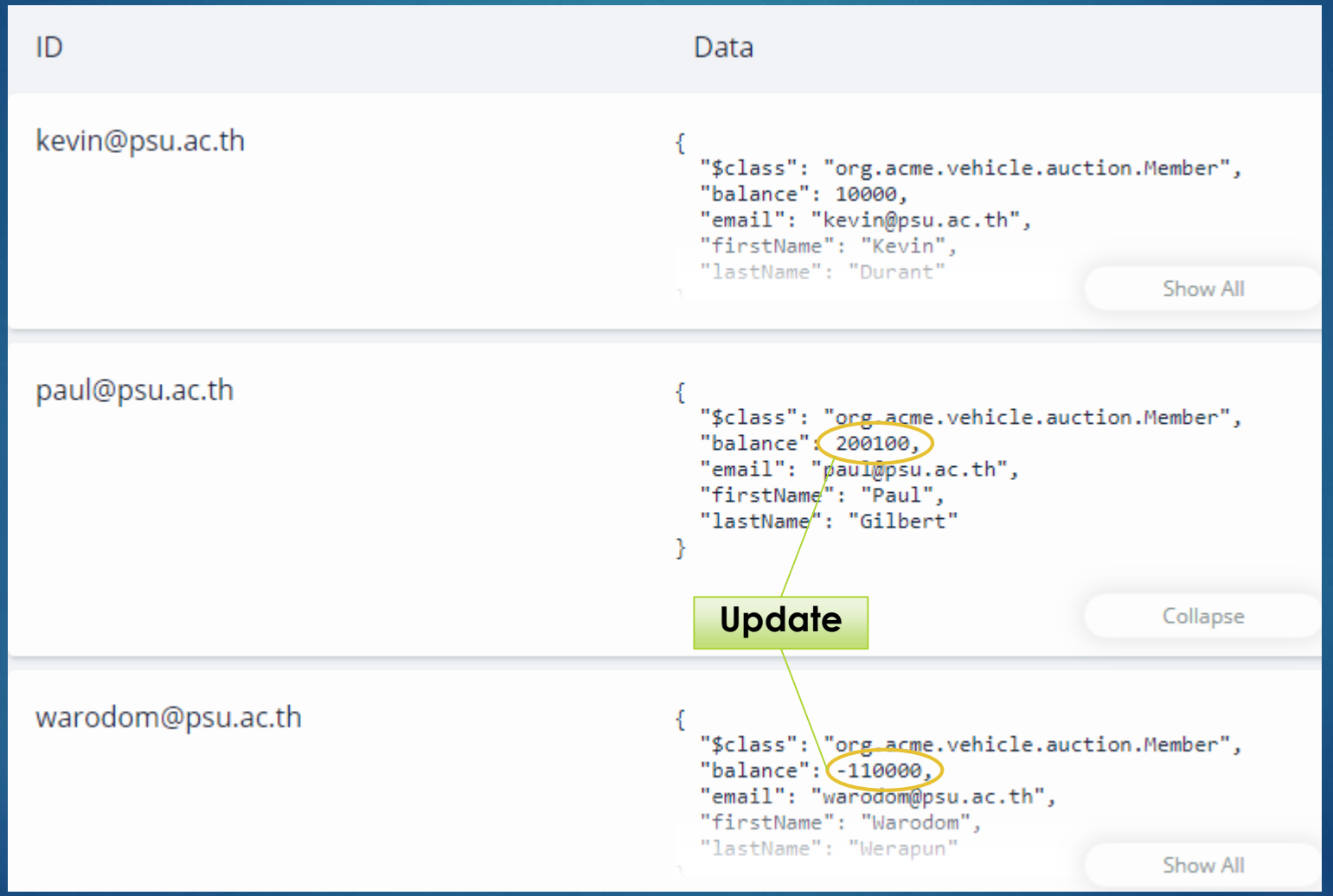
Data

```
{
  "$class": "org.acme.vehicle.auction.Vehicle",
  "vin": "9999",
  "owner": "resource:org.acme.vehicle.auction.Member#warodom@psu.ac.th"
}
```

ID	Data

carListing	<pre>{
	  "$class": "org.acme.vehicle.auction.VehicleListing",
	  "listingId": "carListing",
	  "reservePrice": 1000,
	  "description": "Paul wants to sale his car",
	  "state": "SOLD",
	  "vehicle": "resource:org.acme.vehicle.auction.Vehicle#9999"
	}</pre>

Money is transferred



Check a buyer wallet

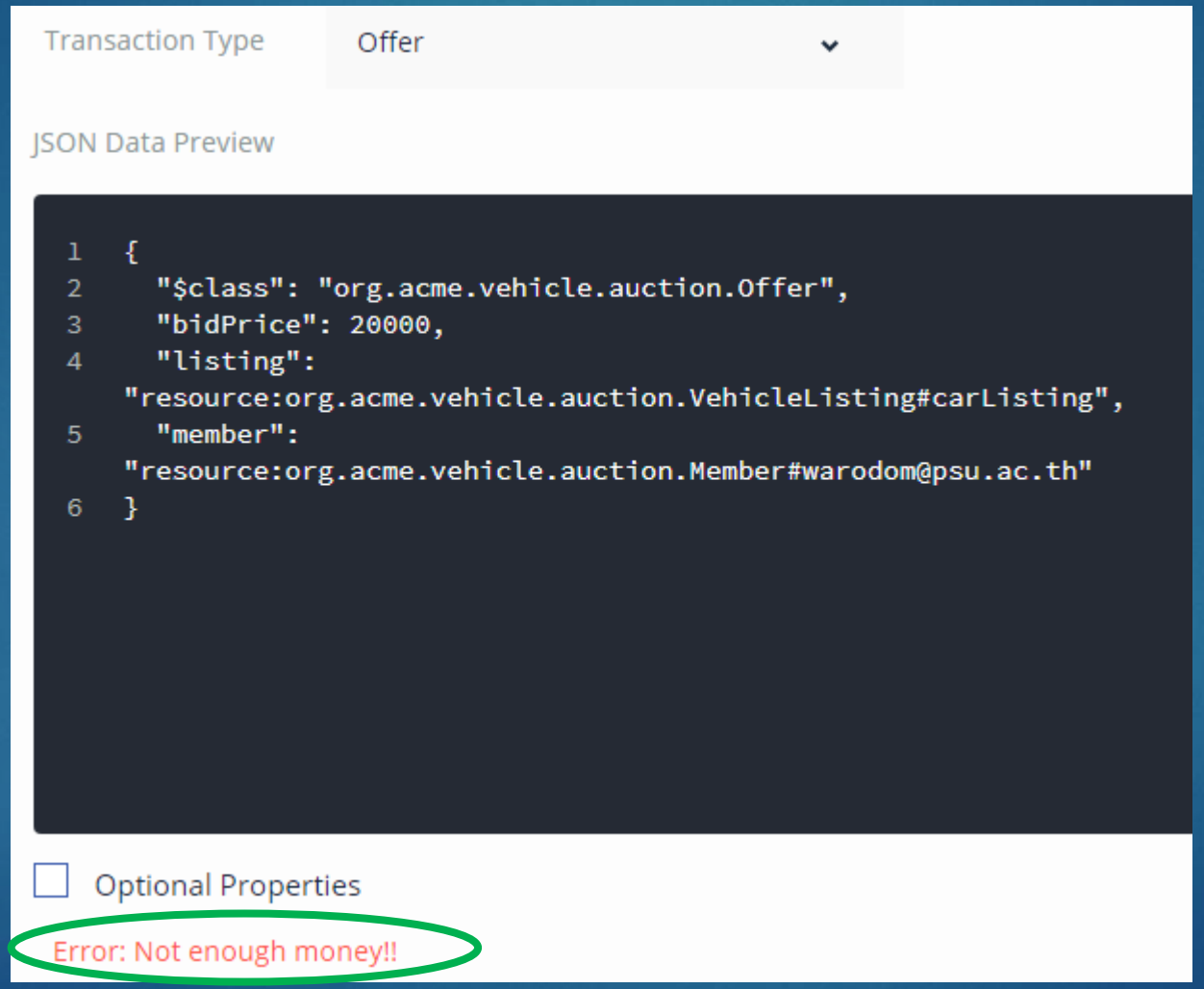

```
async function makeOffer(offer) {  // eslint-disable-line no-unused-vars
    let listing = offer.listing;
    if (listing.state !== 'FOR_SALE') {
        throw new Error('Listing is not FOR SALE');
    }
    if (!listing.offers) {
        listing.offers = [];
    }
    listing.offers.push(offer);

    if ( offer.bidPrice > offer.member.balance )
        throw new Error('Not enough money!!');

    // save the vehicle listing
    const vehicleListingRegistry = await getAssetRegistry('org.acme.vehicle.auction.VehicleListing');
    await vehicleListingRegistry.update(listing);
}
```

Edit lib/logic.js to check **bidPrice** with member **balance**

Not allow to place bidding



References

- https://www.c-sharpcorner.com/article/basics-of-blockchain/
- *https://en.wikipedia.org/wiki/Blockchain*
- https://medium.com/swlh/how-does-bitcoin-blockchain-mining-work-36db1c5cb55d
- https://anders.com/blockchain
- https://hyperledger.github.io/composer/latest/tutorials/playground-tutorial.html