Practical 5 – Blockchain development on Hyperledger Fabric using Composer

Hyperledger Fabric:

→ Hyperledger is an open source collaborative effort created to advance blockchain technologies. The Hyperledger organization has a number of projects for various blockchain solutions, such as smart contract engines, permissioned networks, querying for information inside a ledger, etc.

→ These projects are blockchain related tools built by the open source community and the organization.

→ **Hyperledger fabric** is a permissioned distributed ledger technology.

→ When compared to the widely used applications of blockchain such as Bitcoin or Ethereum which works on public networks and open to anyone, Hyperledger fabric works in permissioned environment.

→ There was a need of maintain identity, recognized networks(permissioned), high transaction throughput, low latency, and maintaining privacy and confidentiality in business transactions for enterprise use cases.

→ It has a highly modular and configurable architecture.

→ Here, the participants are known to each other. This does not translate to the fact that all the participants in a permissioned environment would fully trust each other. They may or may not. But the network is built under a

framework such that any problems can be solved as there exists a co-operative and trust factor between the known participants inside the network itself.

→ This technology could be applied to various use cases such as healthcare, banking, supply chain etc.

→ One more capability that hyperledger fabric has that sets itself apart from other distributed ledgers is that it does not require any cryptocurrency.

Hyperledger Composer:

→ Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier.

→ We can use Composer to rapidly develop use cases and deploy a blockchain solution quickly.

→ Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.

→ Hyperledger Composer quickly models your current business network, containing your existing assets and the transactions related to them.

→ Assets are tangible or intangible goods, services, or property.

→ As part of your business network model, you define the transactions which can interact with assets.

→ Business networks also include the participants who interact with them, each of which can be associated with a unique identity, across multiple business networks.

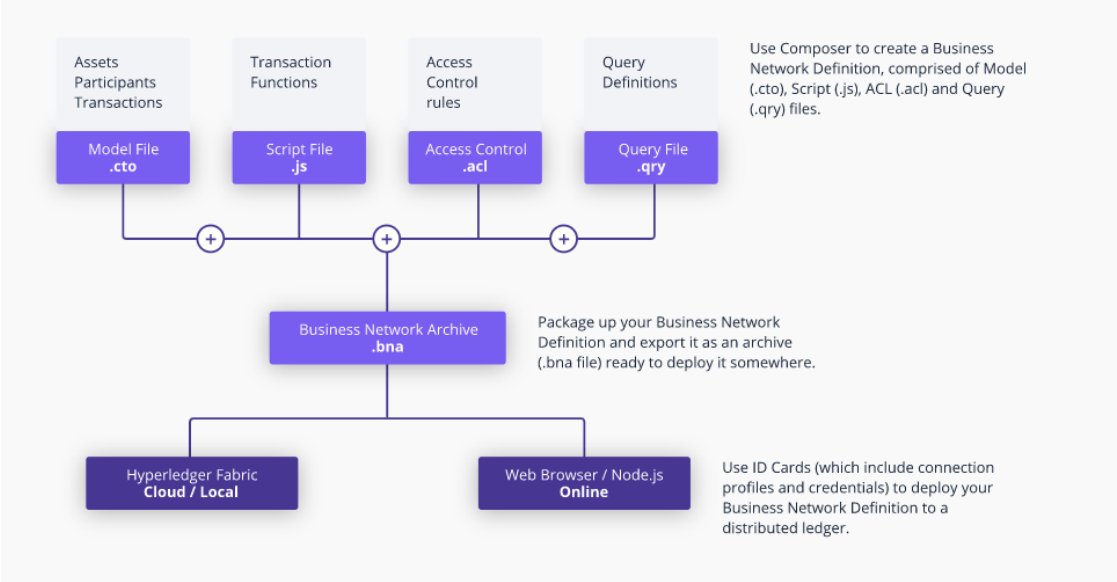

Figure 1: Components of business network archive

→ In implementation we set up a trading cards network where different users (Traders) trade cards among themselves.

→ **Installing prerequisites:**

- Here, we install the necessary prerequisites like docker engine, docker compose, nodejs, npm, git and python.
- The following commands were used:
  - `curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh`
  - `chmod u+x prereqs-ubuntu.sh`
  - `./prereqs-ubuntu.sh`

→ **Installing development tools:**

- Necessary tools for the development of the blockchain are installed.
  - `npm install -g composer-cli`
  - `npm install -g composer-rest-server`
  - `npm install -g composer-playground`
  - `npm install -g yo generator-hyperledger-composer`

→ **Installing Hyperledger Fabric runtime:**

- A runtime environment is setup.
- Also, a participant with userID as PeerAdmin is created with card name as PeerAdmin@hlfv1 is generated as shown below in figure 2.
- The commands used here are:
  - `mkdir ~/fabric-dev-servers`
  - `cd ~/fabric-dev-servers`
  - `curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz`
  - `tar -xvf fabric-dev-servers.tar.gz`
  - `export FABRIC_VERSION=hlfv12`
  - `./downloadFabric.sh`
  - `./startFabric.sh`
  - `./createPeerAdminCard.sh`

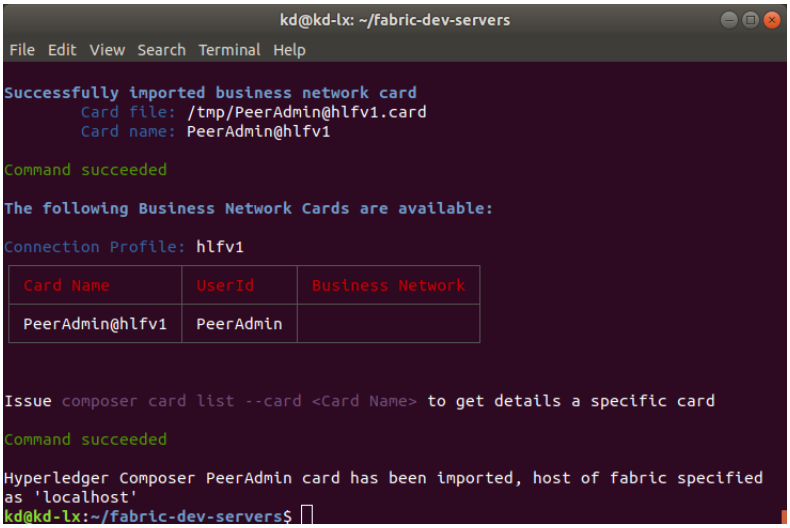

Figure 2

→ **Creating and deploying Business network:**

- Here we will generate project with necessary requirements using the Hyperledger composer project generator which will setup for us sample applications.
- After the business network is generated as using the command “yo hyperledger-composer" we start modeling it.
- A modeling file called as org.example.biznet.cto is generated from previous step.
- In this file we will write a code defining our assets, transactions, events and participants.
- The file will look as shown below after mentioning the necessary resources.

```
/**
* The asset participants will be trading.
* Each card has certain properties such as name,
* description, and type which can
* be used for the frontend application
*/
asset TradingCard identified by cardId {
  o String cardId
  o String cardName
  o String cardDescription
  o GameType cardType default="Baseball"
  o Boolean forTrade
   --> Trader owner
}
/**
* Enumerated types are used to specify a type
* which can have 1 or N possible values, and nothing else.
```

```
*/
enum GameType {
  o Baseball
  o Football
  o Cricket
}
/**
* The participant model for a Trader
*/
participant Trader identified by traderId {
   o String traderId
   o String traderName
}
```

- Now we add the logic behind Tradecard function by setting up a logic.js file in the lib folder inside the workspace directory.
- The following code is written in logic.js.

```
/**
* Buy card transaction
* @param {org.example.biznet.TradeCard} trade
* @transaction
*/
async function buyCard(trade) {
if (trade.card.forTrade) {
// If card is available for trade
 trade.card.owner = trade.newOwner;
return getAssetRegistry("org.example.biznet.TradingCard")
    .then(assetRegistry => {
     return assetRegistry.update(trade.card);
```

```
})
.then(() => {
 let event = getFactory().newEvent(
                "org.example.biznet",
                "TradeNotification"
                 );
   event.card = trade.card;
   emit(event); // Fire off the event
  });
}
}
```

- In order to give permissions to access the resources we define a new rule in permissions.acl file. The following rule is added.

```
rule AllParticipantsHaveAccessToAllResources {
   description: "Allow all participants to have access to all
   resources and make transactions"
   participant: "ANY"
   operation: ALL
   resource: "org.example.biznet.*"
   action: ALLOW
}
```

- We make a business archive file after mentioning all the components that a network requires. So, we deploy the business network in local fabric runtime. The following command is used. Refer figure 3.
  - `composer archive create --sourceType dir --sourceName .`

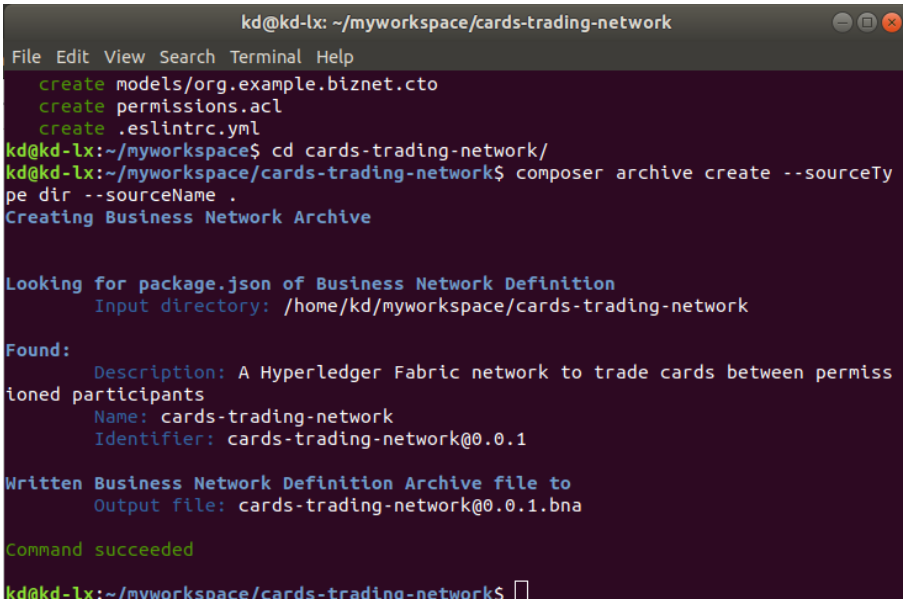

Figure 3: Creating business archive

- Using the PeerAdmin user we created earlier, we install and deploy the network to our local fabric runtime. So, we use two commands here, one for installing the business network and other for deploying it respectively:
  - `composer network install --archiveFile cards-trading-network@0.0.1.bna --card PeerAdmin@hlfv1`
  - `composer network start --networkName cards-trading-network --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file cards-trading-admin.card`

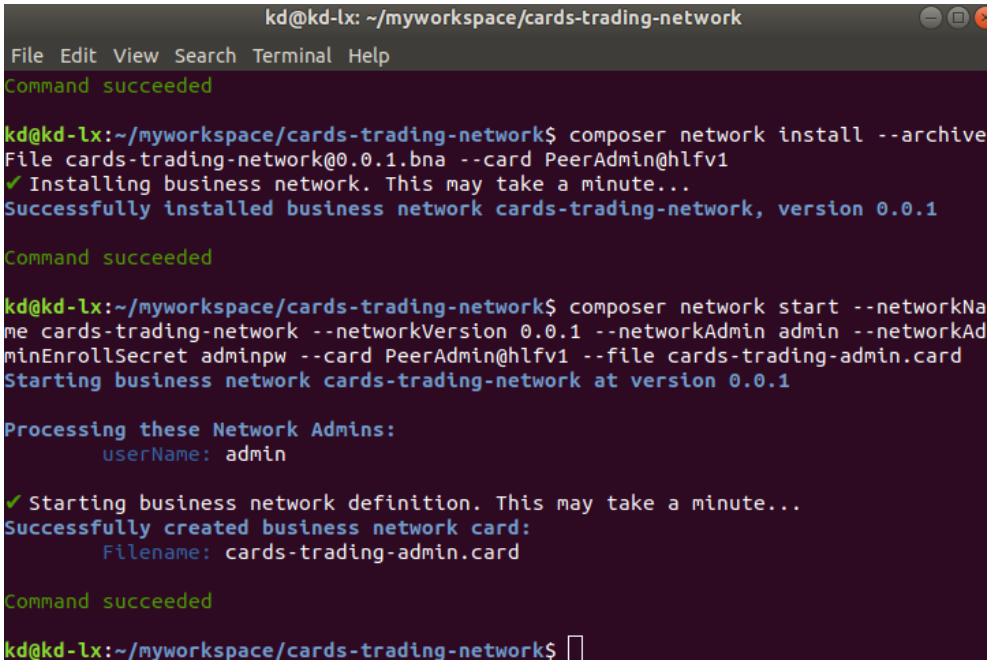

Figure 4: Installing and starting the business network

- We then import our card and confirm that our network is up and running by typing the following two commands. The terminal should look like as shown in figure 5.
  - `composer card import --file cards-trading-admin.card`
  - `composer network ping --card admin@cards-trading-network`

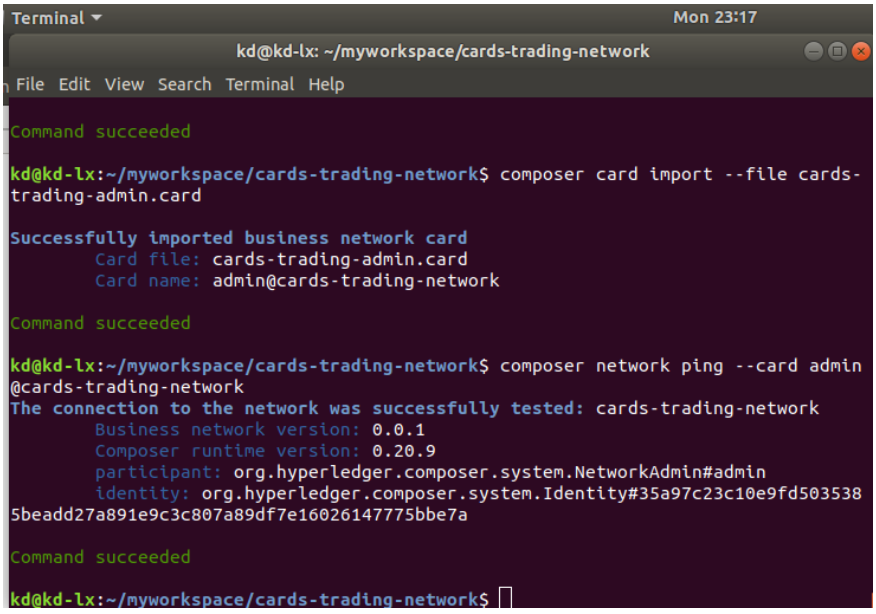

Figure 5: Importing the admin card and making sure the server is up and running

→ **Testing Business network.**

- We use composer-playground to play with the business network we just created.
- We now create new participants in the network one by one as shown in figure 6.

Figure 6: Creating a participant

- Similarly, we create new assets as shown in figure 7.

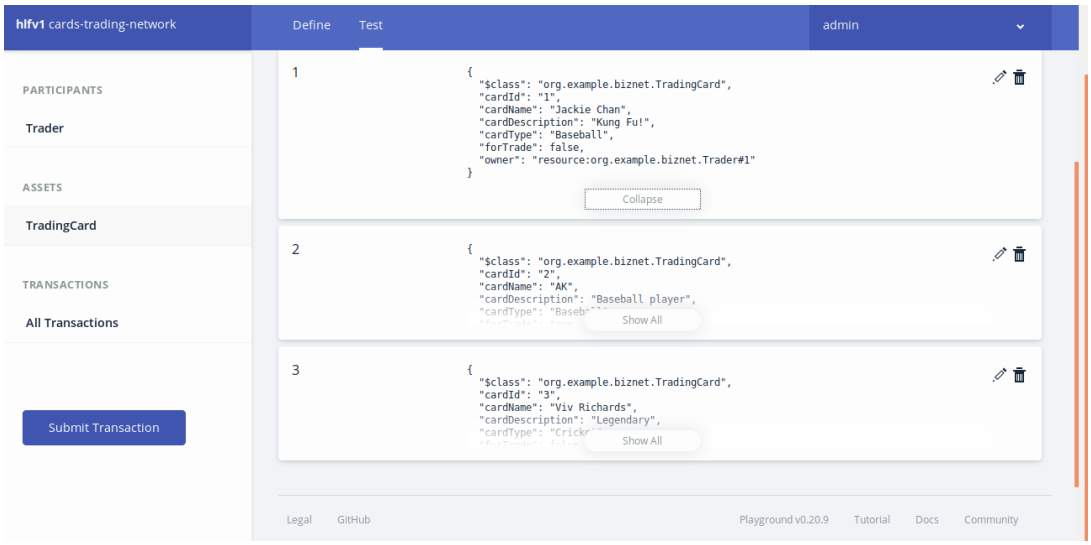

Figure 7: Assets in the network

- Now we click on submit transaction and modify it as shown below.

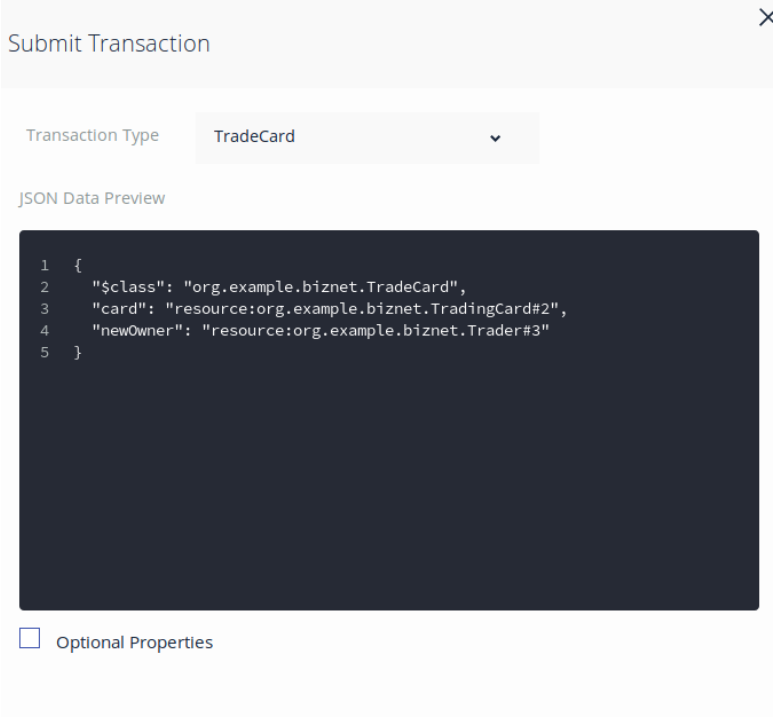

Figure 8: Submitting the transaction

- After we submit the transaction shown in figure 8, we now get the new owner of the card 2 as Trader#3.

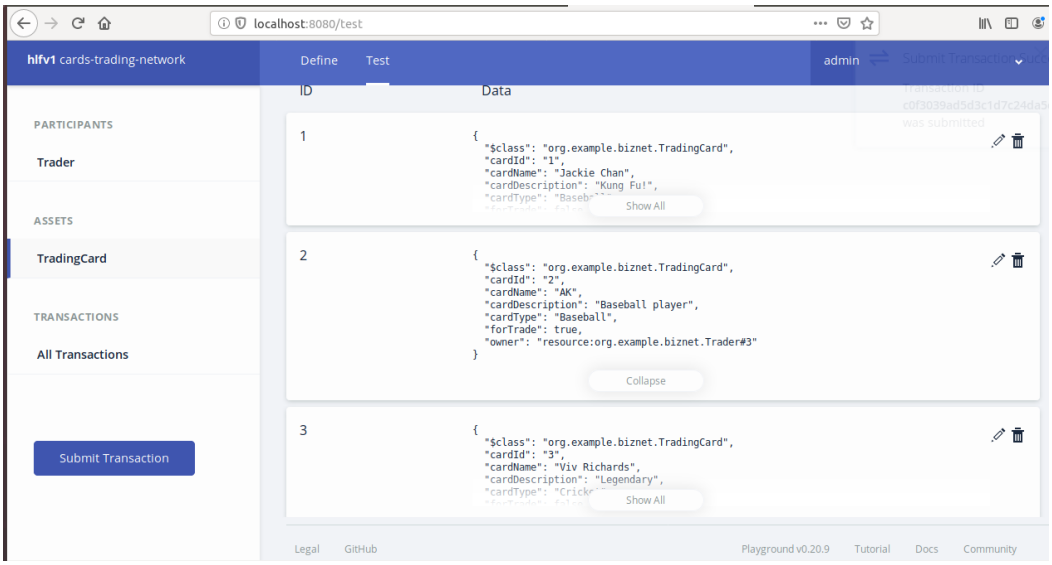

Figure 9: Post trading, the new owner of the card

References

[1] How to build a blockchain network using Hyperledger Fabric and Composer , https://medium.com/free-code-camp/how-to-build-a-blockchain-network-using-hyperledger-fabric-and-composer-e06644ff801d

[2] Hyperledger Fabric Getting Started Tutorial + Installation Guide, https://www.srcmake.com/home/fabric

[3] Welcome to Hyperledger Composer , https://hyperledger.github.io/composer/latest/introduction/introduction