Trading Platform using Hyperledger Fabric

Akhila Katukuri , Sai Jeevan Teegala , Vamshi Krishna
Seidenberg School of Computer Science and Information
Systems Pace University New York, USA

Abstract— Blockchain has emerged as a disruptive technology in the areas of trading assets and sharing information. Hyperledger Fabric and Hyperledger composer are open source projects that help organizations create private, permissioned blockchain networks.Blockchains establish trust across a business network through the combination of a distributed ledger, smart contracts, and consensus.

This paper proposes the various features of Trading platform using Hyperledger Fabric for different traders and their need to exchange their cards with others. It also describes our experience in the design, implementation and architecture of blockchain-based trading networks.
The implementation is based on the permissioned blockchain Hyperledger Fabric.
Keywords—Blockchain,HyperledgerFabric, Hyperledger Composer, Distributed ledger

I. INTRODUCTION

Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.Trading platform is a place where different users who own trading cards of Baseball, Football, and Cricket players come to interchange their cards with others.Different traders who own Trading cards will be able to trade cards among themselves.This can be done by using the set up of REST API server to allow client side software to interact with our business network.We can download the fabric runtime directly using the terminal through npm and start it up. Then we generate a Peer Admin card which is basically YOU the developer. Participants in a Fabric network can have business network cards,analogous to real life business cards.

Fabric is a base layer for private blockchains to build upon. The holder of the Peer Admin business card has the authority to deploy, delete, and manage business networks on this Fabric runtime. Now, we use commands in NPM to generate boilerplate business networks among other things and generate the empty template network structure.

For our trading network , we will define an asset type which is the Trading Card, a participant who is the Trader, a transaction of Trading Cards among themselves and an event of Trade Notification when a transaction occurs.

II. PROJECT REQUIREMENTS

We aim to develop a trading platform where individuals who have trading cards can keep their trading cards for sale and can trade with others. In this project we will use a hyperledger fabric which is an open source framework for making private (permissioned) blockchain business networks. SDK's are available for Nodejs and Java to build client applications. Hyperledger fabric only runs on Unix systems. To build and run this hyperledger fabric there are some prerequisites that need to be installed.

- Docker Engine and Docker Compose
- Nodejs and NPM
- Git
- Python 2.7.x

We also use hyperledger composer for this project. Hyperledger Composer is a set of collaboration tools for building blockchain business networks that make it simple and fast for business owners and developers to create smart contracts and blockchain applications to solve business problems. To develop this project you need not have a full idea of blockchain but need to know the basics of javascript. Hyperledger composer consists of a business network module file where our project is developed.

III. ARCHITECTURE

A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in a single list, called a chain. Blockchains are used for recording transactions made with cryptocurrencies, such as Bitcoin, and have many other applications.

Hyperledger Fabric is a platform for distributed ledger solutions at industrial level. A modular architecture - Delivers high degrees of confidentiality, resiliency, flexibility and scalability. It is designed to support pluggable implementations of different components, and accommodate the complexity and intricacies that exist across the economic ecosystem. Breaks from some other blockchain systems is that it is private and permissioned. Hyperledger Composer is a set of Javascript based tools and scripts which simplify the creation of Hyperledger Fabric networks. Using these tools, we can generate a business network archive (BNA) for our network. Composer broadly covers these components:

- Business Network Archive (BNA)
- Composer Playground
- Composer REST Server

Business Network Archive — Composer allows us to package a few different files and generate an archive which can then be deployed onto a Fabric network. To generate this archive, we need:

Network Model — A definition of the resources present in the network. These resources include Assets, Participants, and Transactions. We will come back to these later.

Business Logic — Logic for the transaction functions

Access Control Limitations — Contains various rules which define the rights of different participants in the network. This includes, but is not limited to, defining what Assets the Participants can control.

Query File (optional) — A set of queries which can be run on the network. These can be thought of as similar to SQL queries.

Composer Playground is a web based user interface that we can use to model and test our business network. Playground is good for modelling simple Proofs of Concept, as it uses the browser's local storage to simulate the blockchain network. However, if we are running a local Fabric runtime and have deployed a network to it, we can also access that using Playground. In this case, Playground isn't simulating the network, it's communicating with the local Fabric runtime directly.

The first and most important step towards making a business network is identifying the resources present. We have four resource types in the modeling language:

- Assets
- Participants
- Transactions
- Events

In this trading platform, we will have an asset type which is the Trading Card , a participant who is the Trader , a transaction of Trading Cards among themselves and an event of Trade Notification when a transaction occurs. Once we open the generated template files if we open the .cto file which is the modeling file. This contains the specification for our asset Trading Card. All assets and participants need to have a unique identifier for them which we specify in the code, and in our case, it's IDs.

Also, our asset has a gametype property for the card. In our project, no Trading card can have a game type other than Baseball, Football, or Cricket. Now, we also specify our Trader participant resource type. We have a participant type Trader and they're uniquely identified by their IDs. By adding a reference to our trading cards we can use this reference to point it to their owners, so we know who the card belongs to. This can be achieved by using the pointer reference.We add logic behind the transaction function using a Javascript logic file, which basically says to start the transaction event when only when the card is ready for trade and another user is trying to buy it. Then, it fires off the trade notification event for that card.

IV. ADVANTAGES

- Permissioned membership
- Performance, scalability, and levels of trust
- Modular architecture supporting plug-in components
- Protection of digital keys and sensitive data
- Time efficient
- Cost efficient
- Enhanced privacy
- Improved auditability
- Increased operational efficiency

V. DISADVANTAGES

- It has shown a lack of use cases.
- It has got a complex architecture.
- It has got minimum APIs and SDKs.
- It is not a Network fault-tolerant.

VI. PRODUCT RESULTS

Select Trader from Participants, click on Create New Participant near the top right, and make a new Trader similar to this:

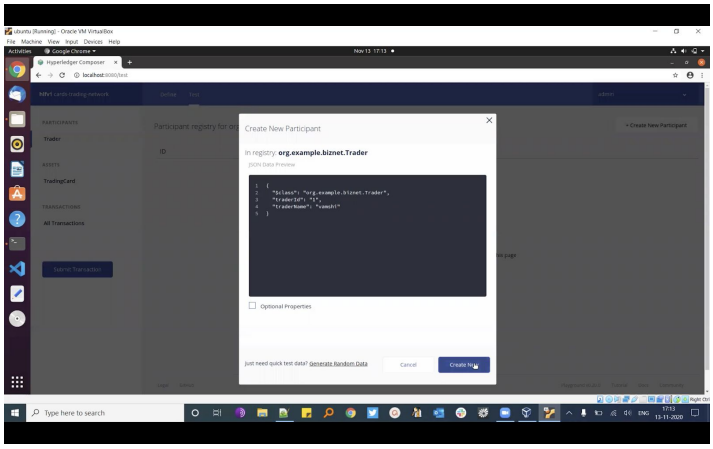

Fig 1: Creating New participant

Go ahead and create a couple more Traders. Here are what our three traders look like with the names Vamshi, Jeevan, and Akhila.

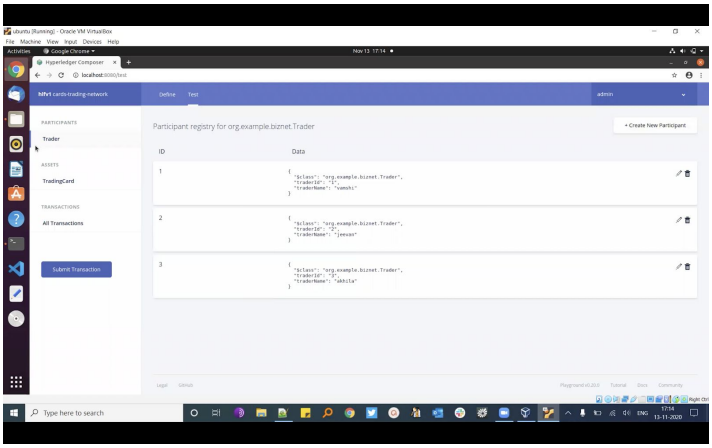

Fig 2: Created Traders

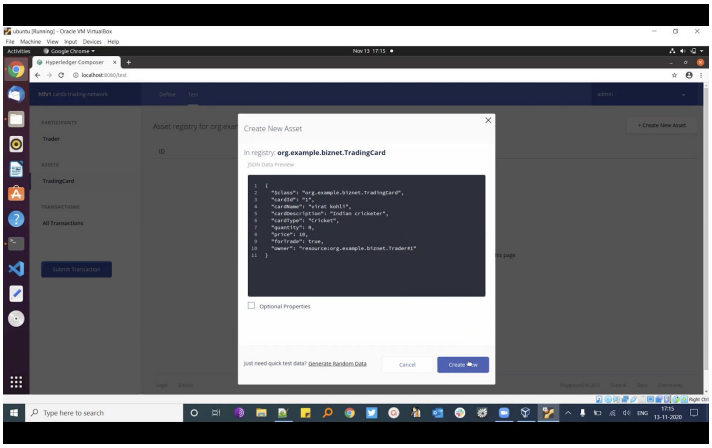

Fig 3: Creating New Asset

Now run npm install , give it a minute, and once it's all done we'll be able to load up http://localhost:4200/ and be greeted with a page similar to this:

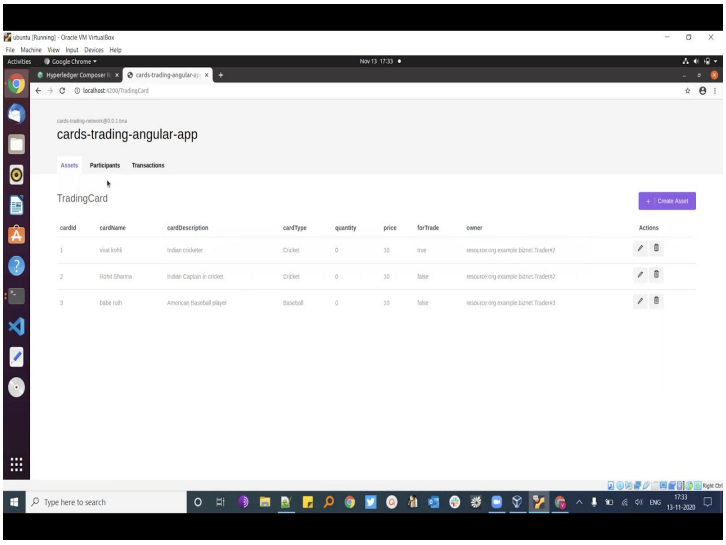

Fig 4: Cards trading angular app

Now invoke and create a new transaction.

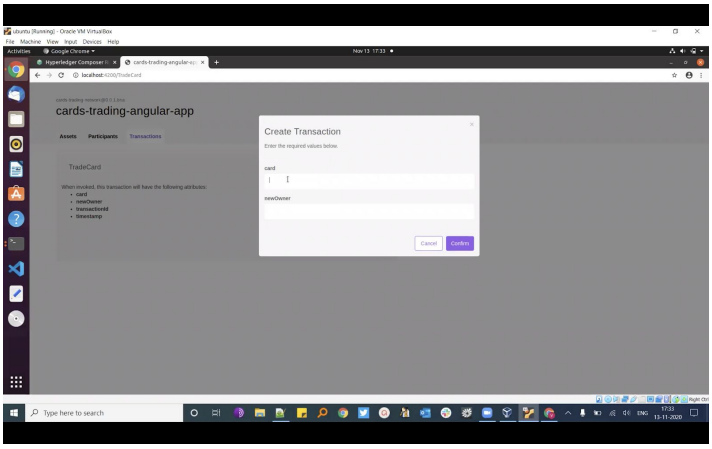

Fig 5: Invoke and create transaction

VII. CONCLUSION

We have successfully created a Trading platform using Hyperledger Fabric with advanced features, where different users who own trading cards can exchange their cards with others securely which includes different resource types such as Assets, Participants, Transactions and Events.

REFERENCES

[1] https://hyperledger.github.io/composer/v0.19/tutorials/developer-tutorial

[2] https://blog.codecentric.de/en/2018/04/blockchain-application-fabric-composer/

[3] https://ieeexplore.ieee.org/document/9034576

[4] https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf