Leveraging IoT and Block Chain Technology in Supply Chain Management

(BCTSCM)

By

Capt Talha Latif

Capt Osama Bin Shabbir

Capt Hamza Saleem

Capt Azmatullah Nasar

Supervised by:

Dr Tauseef Rana

Submitted to the faculty of Department of Computer Software Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad, in partial fulfillment for the requirements of B.E Degree in Software Engineering.

June 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"Leveraging IoT and Block Chain Technology in Supply Chain Management"

is carried out by

Capt Talha Latif

Capt Osama Bin Shabbir

Capt Hamza Saleem

Capt Azmatullah Nasar

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr Tauseef Rana

Date: ___________

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan’Wa’Tala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, **Dr. Tauseef Rana** without your guidance.

The group members, who through all adversities worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has ____ similarity index. Turnitin report endorsed by Supervisor is attached.

Student 1 Name
NUST Serial no

Student 2 Name
NUST Serial no

Student 3 Name
NUST Serial no

Student 4 Name
NUST Serial no

Signature of Supervisor

ABSTRACT

The main objective of this project is to build a Blockchain solution for Supply chain management powered by Hyperledger Fabric. It will contain immutable ledger for all transaction happening during shipping of goods and with the help of these ledger one can get a transparent status of shipping as well as the confidence of delivery date. It can be IOT enable. Android IOT Device App will sense room Temperature and fetch the current GPS location (lat & Long). As of now Application will not have this feature. But we can integrate any Thermocouple sensor bought from Market and can program Android app to get the accurate temperature readings. This data is further pushed into blockchain solution via Node Red- an IOT connector tool by Node.Js

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Overview

A supply chain is a network of organizations that share the goal of creating and distributing high-quality products or services [1]. Before the supply chain was straightforward, where things were produced and transported and later sold in the markets. Now supply chains are much more complex, where the system consists of not only manufacturers and suppliers but also assemblers, warehouse managers, logistics companies for transportation, distributors, retailers, and also consumers [2]. Often, the stakeholders involved have no knowledge of other participants in the supply chain [3], especially the consumer who might not be aware of where, when, or how the products is created.

If a system has three or more organizations or individuals and it can contain upstream and downstream flows of resources it can be considered a supply chain . Organizations can be part of many supply chains at a time which may vary in size and complexity. The typical participants in any supply chain are [4]:

• **Producers:** also known as maufacturers which produce products, it can be raw materials consumed by other maufacturers or final products that is used by customer.

• **Distributors:** also known as wholesalers, buy products from producers in huge amount and sell them to customers. Their main customers are other companies that are interested in purchasing huge quantities than the regular customer. This participant has the role of offering demanded products to customer at the right time in the right place. For this they have large inventories of products and act as a buffer between the customers and producers which protects both parties from fluctuations in demand or supply.

• **Retailers:** They are the one which sells the products to end customers. Their goal is to make the product attractive to buy by combining marketing strategies and service.

• **Customers:** they buy and use products. They are either end consumer or another type of supply chain participant that uses the bought product for the creation of other products which in turn get sold to other customers.

Supply Chain Management (SCM) encompasses the planning and management of all activities involved in sourcing and procurement, conversion, and all logistics management activities. Importantly, it also includes coordination and collaboration with channel partners, which can be suppliers, intermediaries, third-party service providers, and customers [5]. SCM involves the management of products, , and funds. Because of globalization, SCM is now prone to risks due to the various interconnected parties, and a minor fault could result in negative consequences for all members [6]. Risks include theft, counterfeit, raw material price fluctuations, natural disaster, logistic delays, environmental hazards, economic instability, and supplier inconsistency [7]. Out of these, counterfeit and fraud account for 11% of the total risks [7].

1.2 Problem Statement

The result of using counterfeit goods end in a considerable reputation, financial, material, and, in the worst case, life loss. It can be countered by giving a Traceable Product History without disclosing business secrets. A Traceable Product History is the chronological sequence of steps taken in a supply chain. The steps are actions taken by a member in the supply chain that change the state of the product, including submitting an order, acquiring a raw natural resource (materializing as a product), testing, altering a component, combining two or more products into a single one, transfer in ownership or responsibility. This product becomes now an asset and each step is to be considered as a transaction on this asset with data (information) associated with it. Transactions and relevant information are traditionally stored in volatile documents that can be lost, un/intentionally omitted, or tampered with and need to be continuously duplicated and manually synced between parties. A client's inquiry on a product can take weeks to be consolidated because data needs to be gathered from multiple sources and parties via the mentioned inefficient data flow. Data flow occurs during a transfer of ownership or change of status between the two parties. In worse cases, disputes between parties can lead to third parties' involvement to resolve a claim.

Current supply chain management systems allow to automate the monitoring and collection of information related to the various activities within the supply chain. In this way, the set of traceability information of a product allows the future consumer to know the provenance of that product and the events related to its entire life cycle

However, most of today's supply chains are managed by centralized systems: members of such supply chains rely on a centralized authority, more specifically an information supervision center, to transfer and share their information. These centralized systems are often non-transparent, monopolistic and asymmetric information systems. This can pose a serious threat to the security and reliability of the traceability information and make fraud, corruption and data falsification easier. Furthermore, such centralized systems have limited scalability and a single point of failure.

1.3 Proposed Solution

These problems can be solved by a transparent data flow, which can be achieved by having a tamper-proof single data source such as a distributed ledger technology, a blockchain-based system that keeps track of records, product information, and history. Such system treats the asset and transactions as digital ones. A client accessing data from a blockchain can easily verify a product's authenticity and detect counterfeit products. Depending on the blockchain used, systems can be implemented to provide different access levels for their users. In addition to transparency, the blockchain based system offers accountability and scalability through multiple organizations and locations. Adopting blockchain could help the supply chain industry reduce risks, improve management, and increase consumer and manufacturer trust.

Blockchain technology in particular offers cryptographic primitives to store data within a distributed ledger, guaranteeing their immutability and authenticity. This eliminates the need for supply chain members to trust a single entity to manage their traceability information. Furthermore, being a distributed system, the blockchain can solve the problems of limited scalability and single point of failure.

The proposed system in particular allows supply chain members to store and manage product-related traceability information in a distributed and immutable way. An important component of this system is the smart contract, a blockchain primitive that allows to automate some of the management operations related to supply chain activities.

1.4 Objectives

The main objective of this thesis is to apply permissioned blockchains technology namely Hyperledger Fabric to supply chain management.

1.4.1 General Objectives:

- Improved collaboration between the parts involved in the supply chain;
- Transparency in every stage;
- Improved security and encryption;
- Decentralization of data and the possibility to reach consensus;

1.4.2 Academic Objectives:

- Development of a reliable and user-friendly SCM system
- To implement the knowledge learned during the BS degree
- To increase productivity by working in a team
- To design a project that contributes to the welfare of society

1.5 Scope

This project finds its scope in all industrial areas where single or multiple originations are working together to manufacture and deliver products to consumers. The benefits of this project as mentioned above will also help the consumers.

1.6 Deliverable

1.6.1

A Fabric permissioned blockchain network is a technical infrastructure that provides ledger services to application consumers and administrators. In most cases, multiple organizations come together as a consortium to form the network and their permissions are determined by a set of policies that are agreed to by the consortium when the network is originally configured. Moreover, network policies

1.6.2 Chaincode with GoLang

Chaincode, or a smart contract is a fragment of code written in supported languages like Java or Go that is deployed onto a network of HyperLedger Fabric peer nodes. Chaincodes run network transactions which are validated and then appended to the shared ledger. In simple terms, they are encapsulation of business network transactions in a code.

1.6.3 Middleware APIs using Node Js

Application programming interfaces (APIs) are everywhere. They enable software to communicate with other pieces of software (internal or external) consistently, which is a key ingredient in scalability, not to mention reusability.

1.6.4 Frontend app with React.js

The frontend of a web application is the part the user interacts with directly. It is usually referred to as the application's "client side." The frontend consists of everything that the user sees when interacting with the website or app, such as text colors and styles, photos, graphs and tables, buttons, colors, the navigation menu, and much more. Frontend developers provide the structure, appearance, behavior, and content of everything that appears on browser displays when websites, online applications, or mobile apps are opened. The key focus points of frontend development are responsiveness and performance.

1.7 Relevant Sustainable Development Goals

This app wouldn't just improve the practical day-to-day operations. Once in place, this robust, agile platform would help users mitigate damage from unpredictable global disruptions, reduce costs and make more profit due to higher efficiencies and end-to-end visibility of their data.

- Increase Aid for Trade support for developing countries

- Increase the access of small-scale industrial and other enterprises
- Enhance scientific research, upgrade the technological capabilities of industrial sectors
- Support domestic technology development, research and innovation in developing
- Promote the rule of law at the national and international levels
- Ensure ethical sourcing through transparency.
- Minimize overproduction through efficient supply and demand planning.

1.8 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2: Literature Review

A new product is launched by modifying and enhancing the features of previously launched similar products. Literature review is an important step for development of an idea to a new product. For a better understanding of the project contributions, it is important to introduce some background elements related to the concepts and paradigms used. First it will start by describing the blockchain technology from a wider perspective, history and types of blockchains. In the second part we will dive a bit deeper into the permissioned Hyperledger Fabric network and the specifics about it.

2.1 Blockchain

The beginning of blockchain technology is marked by the introduction of Bitcoin by notoriously famous, but still unknown group or individual Satoshi Nakamoto in [8], where a peer- to -peer electronic cash system is proposed, which allows sending payments between parties without a central authority. Several years later, Bitcoin represents a multibillion dollar market with a multitude of varieties and adaptations of the original network. Currently the world relies on third party actors whose main role is to offer security, provide confirmation of our online actions or certify identities. The presence of these third parties in every online process makes systems more complex, opening more possibilities for security vulnerabilities, fraud, corruption and manipulation. Blockchain has the advantage of being able to remove these limitations by introducing a distributed consensus. Moreover it does so by preserving the privacy and anonymity of involved parties. With such advantages at hand, the current way of how things work can be challenged, putting a new system in place with fundamental, game- changing differences. This potential outweighs all the complexity, regulatory and technical challenges that come with this technology.

2.2 Blockchain Architecture

A blockchain essentially represents a distributed database containing records, transactions or events information that are shared among all the participants of the network [9].

2.2.1 Blocks

Every block has only one parent block, to which it is linked through a hash. The only exception is the first block recorded on the ledger, which has no parent and is called genesis block. A block consists of three sections [10]:

1. **Block Header** consisting of the following fields which are cryptographically derived internally:

- Block Number representing the sequence number of the current block since the creation of the genesis one, which has the 0 index.
- Block Hash, which is made of all the contained transactions from a block, hashed together.
- Previous Block Hash, which points to the anterior block header hash. In this way blocks are linked between them.

2. **Block data** containing ordered transactions

3. **Block Metadata** containing identity information of the block creator and used for validating the blocks by committers

2.3 Blockchain Implementations

The blockchain is implemented as permissionless such as Bitcoin [11] and permissioned such as Hyperledger Fabric [12]. On one hand, permissionless blockchains do not have any control on transactions read/write accesses. On the other hand, permissioned blockchains have integrated mechanisms to control users' transactions read/write accesses, and these accesses are granted to a limited and known number of users.

The blockchain could be deployed in public, consortium, or private environment [13].

• **Public blockchain:** read and write access in the blockchain are open to everyone. Any user in the world can join the network, read, and write transactions in the blockchain. It also guarantees the user's anonymity.

• **Consortium blockchain:** a list of pre-selected nodes control read and write access in the blockchain. It is useful in a context of multiples organizations with data secure share needs, as in the B2B logistic chain context.

• **Private blockchain:** only one organization controls all the rights in the blockchain

2.4 Smart Contracts

As stated by [14], smart contract designates the hard coding of all contract clauses in a hardware or software to be executed automatically in a secured and distributed environment. In the blockchain, the smart contract are self-executed programs that run on the top of blockchain. They are used to develop business application on the top of the blockchain. Bitcoin proposed a script language for smart contract development; however, it was very limited. In the end of 2013, Ethereum [15] came with an integrated framework for smart contract development. Since, it has become a standard in blockchain implementations to integrate the support of smart contracts. The recent permissioned blockchains support smart contracts development in many of wide used languages such as Java in Hyperledger Fabric [12] and Python in Hyperledger Sawtooth [16]. In the logistic domain, the smart contracts could be used to write the business rules agreed between the logistic chain stakeholders. Hence, these rules will be shared and executed transparently among all the logistic chain stakeholders.

2.5 Hyperledger

Hyperledger is an open-source project started by The Linux Foundation that houses several blockchain platforms, jointly developed with other companies (Hyperledger Architecture Working Group, 2017).

HyperLedger cultivates and promotes many Blockchain technologies related to the business aspect, which include smart contract engines, distributed ledger frameworks, client libraries, graphical interfaces, utility libraries, and sample applications. The HyperLedger umbrella design inspires the re-use of common building blocks and enables rapid development and innovation of Distributed ledger technology components.

There are six HyperLedger Frameworks. Out of which only two has active status and rest are in

1. Burrow

2. Fabric

3. Grid

4. Iroha

5. Indy

6. Sawtooth

2.6 Hyperledger Fabric

Hyperledger Fabric falls within the category of permissioned blockchains [17]. In a permissioned blockchain, only authorized peers can participate in blockchain operations. Hyperledger Fabric is a distributed operating system that runs applications written in general purpose programming languages, such as Go, Java, JavaScript, and Python. It introduces the execute-order-validate blockchain model for transaction processing unlike other traditional blockchain systems that use the order-execute model

2.6.1 Components of the Hyperledger Fabric

1. **Ledger :** A ledger consists of two distinct, though related, parts — a “blockchain” and the “state database”, also known as “world state”. Unlike other ledgers, blockchains are immutable — that is, once a block has been added to the chain, it cannot be changed.
2. **Membership Service Provider :** The Membership Service Provider (MSP) refers to an abstract component of the system that provides credentials to clients, and peers for them to participate in a Hyperledger Fabric network.
3. **Smart Contract :** A smart contract is code — invoked by a client application external to the blockchain network — that manages access and modifications to a set of key-value pairs in the World State.
4. **Peers :** A network entity that maintains a ledger and runs chaincode containers in order to perform read/write operations to the ledger. Peers are owned and maintained by members.

5. **Ordering Service :** A defined collective of nodes that orders transactions into a block. The ordering service exists independent of the peer processes and orders transactions on a first-come-first-serve basis for all channel's on the network.
6. **Channel :** A channel is a private blockchain overlay which allows for data isolation and confidentiality. A channel-specific ledger is shared across the peers in the channel, and transacting parties must be properly authenticated to a channel in order to interact with it.
7. **Certificate Authority :** Hyperledger Fabric CA is the default Certificate Authority component, which issues PKI-based certificates to network member organizations and their users.
8. **Organizations :** Also known as "members", organizations are invited to join the blockchain network by a blockchain service provider. An organization is joined to a network by adding its Membership Service Provider to the network.

2.6.2 Identity management

There are various entities in Blockchain network like peer, orderers, applications, administrators. Each of these entities needs to have a valid identity in the network. In HyperLedger each user has an X.509 certificate which serves as a valid identity for each entity. These certificates play a key role in determining the permission access to various resources in the network. A certificate is as trustworthy as the genuineness of the authority it is coming from. A Membership Service Provider (MSP) is used for specifying the rules which identify the valid identities for the various entities.

2.6.3 Public Key Infrastructure PKI

PKI is used for providing secure communications in a network. It makes sure each entity in a network is valid. By default, fabric implements MSP which uses PKI (Public key infrastructure) and X.509 certificates for the entities. A PKI consists of the following elements:

1. Certificate Authorities

2. Public and Private Keys

3. Certificate Revocation Lists

4. Digital Certificates

2.6.4 Digital Certificates

A digital certificate could be seen as analogous to a person's ID/Driving License card. It contains a set of attributes which helps us identify the owner of the digital certificate. A digital certificate also includes the public key of the entity as well along with other relevant information whereas the private key is kept secret and never included in the digital certificate. All this information is encoded using the cryptography. Modifying or tampering of the details in the certificate will make the certificate invalid. In HyperLedger we use the X.509 standard for a digital certificate. As long as Certificate Authority CA 's identity is not compromised (i.e its private key is kept secret) we can be sure that the certificate of the entity has not been tampered.

2.6.5 Public Keys and Private keys

The public key is the key which entity distributes along with the digital certificate. The private key is kept secret. The relationship between public and private key is such that a message signed by the entity's private key can be verified by the entity's public key. The message signing using private key also ensures the message integrity as well as message authentication.

2.6.6 Certificate Authority

Certificate Authority widely distributes certificates to various entities. All the nodes in the Blockchain need to have a valid digital identity which may be distributed by one or more CA. The CA keep its private key as well as the entity’s private key secret. CA makes its certificate publicly available which includes its public key. This makes entity to verify to each others certificate as a certificate signed by CA’s private key can be verified by CA’s certificate which includes its public key

2.6.7 PKI Vs MSP

A PKI provides a list of valid identities in the network. An MSP, in contrast, tells us what identities out of the valid identities provided by PKI forms a part of the Blockchain network. In other words, MSP is the one controlling the valid identities in the Blockchain network.

2.7 Hyperledger Blockchain configuration

The HyperLedeger configuration is the heart of the Blockchian network. We need to configure various entities peers, organizations, policies, channels and a lot more. Most of the HyperLedger configurations goes into the yml or yaml files. HyperLedger provides the following binaries for generarting various types of configurations:

• configtxgen This toll is used for generating the first block of the ledger also known as the genesis block. This block is necessary bootstrapping the orderer as well as the channel configurations.

• cryptogen It is used for generating the crypto material for various entities of the organization

2.8 Use of dockers and network architecture

We use docker containers[18] for configuring various entities in the Blockchain Network. For our Blockchain network we have containers for the following entities:

• Peer containers

• Fabric CA (Certificate Authority) containers

• cli container to access all the peers in the network.

• Orderer container

• couchdb containers

All these containers are a part of a single docker network.

2.9 Block structure and Blockchain structure

A HyperLedger Blockchain record or ledger comprises of two particular yet related components:

• A world state

• A Blockchain

2.9.1 World State

World state is a database that stores the current values of various states of the Ledger. Current values of these states can be easily read from the World state which avoids traversing the entire transaction log and thus saves time. Record or ledger states are represented as key-value pairs which are updated, created and erased frequently.

2.9.2 Blockchain

It is a log of all the transactions that lead to the world state. These transactions are organized into blocks which are linked together to form the Blockchain enabling us to have a record of all the history that led to the current world state. Unlike world state, it contains immutable blocks which contain a set of ordered transactions.

2.10 Why Hyperledger Fabric

We build the application on Hyperledger Fabric for the following reasons:

• Because of the private and permissioned nature, it is ideal for the supply chain application as only its members should have access to the system, and the network can be run by participating nodes.

• Does not require a cryptocurrency to execute transactions, therefore reduces costs when the application scales.

• Because there is no proof of work consensus algorithm, the process of validating transactions is much faster, which is suitable for enterprise cases.

• Channels can be used to partition the data between different domains of the supply chain, and in this way an organization can be part of a channel together with his suppliers, and at the same time part of another channel with his buyers.

Chapter 3: System Design and Implementation

In this chapter we will dive into the system design process and will try to explain the design choices that were made. This will be done by describing each component individually, their roles and challenges associated with them. We will start by describing the most important element of the system, which is the Hyperledger chaincode, followed by the REST server and finally we will talk about the front-end application.

In the traditional supply chain models, the information about an entity is not fully transparent to others, which leads to inaccurate reports and a lack of interoperability. Emails and printed documents provide certain information, but still can’t contain fully detailed visibility and traceability information since the products across the entire supply chain are hard to trace. It is almost impossible for the consumer to know the true value of the product they purchased. Since the blockchain is a transparent, immutable, and secure decentralized system, it is considered to be a game-changing solution for traditional supply chain industries. It can help to build an effective supply chain system by improving the following areas:

- Tracking the products in the entire chain

- Verifying and authenticating the products in the chain
- Sharing the entire chain information between supply chain actors
- Providing better auditability

3.1 System Overview

3.1.1 Flow Description:

The system uses a dynamic way of finding the path for a shipment and assumes organizations can physically send packages to any organization unless mentioned otherwise in the policy. Organizations buy available items from other organizations and pay for them off-chain. After the buying contract is created by the buyer organization, the seller gets notified through a Hyperledger Fabric event, after which he creates the delivery shipment attaching desired delivery rules. The way the shipment gets delivered through the supply chain is that each node is responsible for sending it to the next destination, by following the specified constraints in the policy.

After the delivery shipment is created, the system filters all available organizations based on the attached policy. Out of those filtered organizations, one is chosen as the next stepe. In a supply chain, the type and number of participants can vary greatly from the supplier of producer to the final customer. For this system, a simplified version of supply chain is adopted. The main stakeholders are:

• Manufacture

• Wholeseller

• Distributor

• Customers

These 4 actors can generally represent the majority of supply chains. Even though the customer is also part of the supply-chain, it does not take part in the system as a full member, because it only consumes what the supply-chain has to offer, and for the sake of simplicity. In the current supply chain, the flow of good would start at the producer and end at the customer.

3.1.2 Application Flow :

- Users are enrolled into the application by an Admin.
- New Products will be created by the Manufacturer only.
- Then Product will be sent to WholeSeller
- Wholesalers will send the product to the Distributor.
- Distributor will send it to Retailer
- Consumer could place the order
- Consumer will mark as Delivered once the product is delivered

3.1.3 Overall system architecture

Our Hyperledger Network will consist of

1. Three Orgs

    a. Manufacturer (Org 1)

b. MiddleMen (Org 2)

c. Consumer (Org 3)

2. Five peers
3. One Orderer (Org 1)
4. One Channel
5. Three Certificate Authority (Each org will have a CA)
6. Fabric network implemented with Fabric CA as certificate authority.

3.1.4 System Requirements

The system developed for this project is blockchain based and should maintain an immutable list of past transactions. Transactions are records of events that occur in the blockchain with a given timestamp and are generated using smart contracts. Smart contracts are business code that can modify the state of the ledger. Only authorized parties in the network can access the ledger and execute the smart contracts using API calls.

3.1.5 Business Requirements

The Supply chain system is divided into two modules, the Transparent Data Flow module and the Product Tracking module.

3.1.6 Chaincode Functions

- createUser (Admin)
- signIn (user Login)

- createProduct: (Manufacturer)
- updateProduct (Manufacturer,Wholesaler,Distributor,Retailer)
- sendToWholesaler
- sendToDistributor
- sendToRetailer
- sellToConsumer
- QueryAsset (Query by Product ID)
- QueryAll (All)
- orderProduct(Consumer places order , productID -> Retailer)
- deliveredProduct (Retailer Updates)
- Init - Initialize Counters to NIL
- Invoke - To invoke each function in the chaincode

3.2 Tools and Technologies

3.2.1 Hyperledger Fabric

Hyperledger is an open-source community focused on creating frameworks, tools, and libraries for enterprise blockchain developments [19].

3.2.2 Docker

Docker is an open-source container technology that allows developers to package applications with the dependent libraries [20]. The docker tools are designed to help developers create, run, and deploy applications quickly. The developers need not worry about the system environment that the project would run on. Docker containers can be deployed anywhere, e.g., virtual machines, cloud, etc.

3.2.3 React.js

React.js is an open-source JavaScript library used to implement UI components for web apps developed and maintained by Facebook. Some of the distinguishing features are Virtual DOM, Unidirectional data flow and is light-weight.

3.2.4 Node.js

Node.js is an open-source JavaScript runtime environment build of Chrome's V8 JS engine [21]. Rather than having multiple languages to implement the front end and backend for a web app, node.js unifies the development into one language. For this thesis, we use Node.js as a backend technology to run the server hosting the APIs that communicate with the hyperledger network.

3.3 Process description

The implementation process of the whole system consisted of three steps:

• Blockhain system implementation - started with building a Fabric test network, that would be able to deploy and invoke chaincodes. Then it followed by modifying the built network by modifying names and adding an extra node in order to get the desired network topology and accommodate the design. After the network was deployed, the development of the chaincode has begun by implementing the designed system specification. Functionalities were being tested from the terminal while they were being implemented. The process was considered finalized once it was possible to perform all the action flows by transaction invocation.

• REST Server implementation - the design and implementation of the system came as a necessity after finding out that client applications can not connect to the Fabric network. This component implements only the methods that are necessary by the front-end side. Therefore the process consisted by outlining the needed functionalities, implementing them one by one and testing them with CURL tool.

• Front-end implementation - started with setting up a template application, followed by building the main empty components. Then the pages were being developed one by one until the action flows were possible to complete from the application.

3.4 Network Deployment Prerequisites

The network setup consisted in following the test network setup tutorial from the Hyperledger Fabric documentation page [22]. For this, a series of prerequisites are necessary to be installed, such as

Prerequisites

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (Note: version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher

- Python: 2.7.x
- A code editor of your choice, we recommend VSCode.

This is followed by cloning the Hyperledger fabric-samples [23], which contain the necessary binaries and pre-made projects that can help in the learning process of Fabric. One of the most important things from the fabric-samples repository are the platform-specific binaries and configuration files which are saved in the /bin and /config directories of fabric-samples. Finally, the Hyperledger Fabric docker images are pulled.

3.4.1 Setup

this project was built on top of the Fabric 2.0 test-network project by modifying the necessary configuration files and scripts in order to adapt the network to our needs. Every node is running in a separate docker container, just as the deployed chaincodes. The network setup is strongly dependent on several configuration files. Among them are:

• configtx.yaml, which has channel transaction files and configuration information for building the genesis block. It is split in several sections:

– Organizations - which hold the details about individual organizations and hold their identities that are referenced in the rest of the file. The identities contain the organization name, MSP ID, the directory that contains the MSP configuration and the read, write, admin and endorsement policies for this organization. Finally, the host and port of the anchor peers or orderer endpoint are specified, which get encoded in the genesis block. They are used for communication between nodes in the gossip protocol. Here we define one orderer organization OrdererMSP and three peer organizations: ProdMSP, DistMSP and RetMSP.

– Capabilities - which define the capabilities of the Fabric network. These receive the profiles defined in the next sections.

– Application - application defaults.

– Orderer - the details regarding the orderer parameters. This includes the orderer implementation configuration, TLS certificates or block property configurations.

– Channel - define the values to encode into a config transaction or genesis block for channel related parameters.

– Profiles - which describe the organization structure of the network. Here we specify that the consortium is composed of the producer, distributor and retailer organizations and what kind of capabilities does the orderer have. Also, we define the channel configuration, capabilities and which organizations are part of it. For this proof of concept, one channel for the whole network is considered, as the network is composed only of three peers and there is no reason in having separate channels.

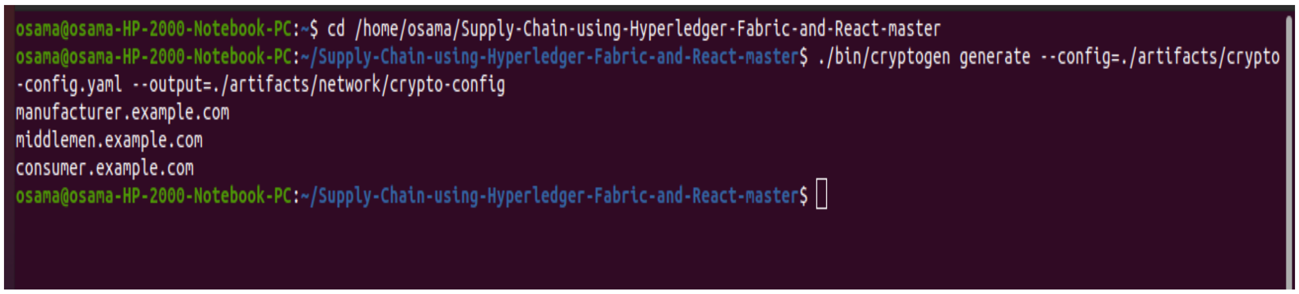
• crypto-config-*.yaml, that has the orderer and peer informations for generating their certificates. These files are required only if cryptogen tool is used for certificate generation. Here the organizations are defined with their name, domains and number of users.

• fabric-ca-config-*.yaml, that has the Certificate Authorities parameters. These files are required only if the organizations' certificates are using certificate authorities instead of the default cryptogen tool.

• docker-compose-*.yaml, these files contain the docker configurations of the containers, such as addresses, domains, ports, path to the genesis block, chaincode ports and other variables important for a correct network setup. These files are also important for setting up the certificate authorities containers.

The creation and deployment process of the network is done by executing following commands using shell script.

1. generate certificates
../bin/cryptogen generate --config=./artifacts/crypto-config.yaml --output=./artifacts/network/crypto-config

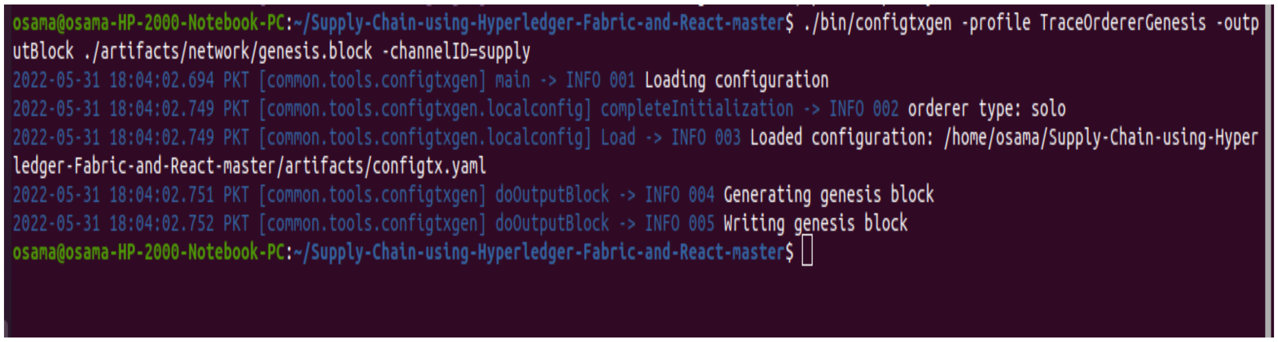


```
osama@osama-HP-2000-Notebook-PC:~$ cd /home/osama/Supply-Chain-using-Hyperledger-Fabric-and-React-master
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/cryptogen generate --config=./artifacts/crypto
-config.yaml --output=./artifacts/network/crypto-config
manufacturer.example.com
middlemen.example.com
consumer.example.com
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ 
```

Fig 1

2. export
export FABRIC_CFG_PATH=${PWD}/artifacts

3. generate the genesis block
../bin/configtxgen -profile TraceOrdererGenesis -outputBlock ./artifacts/network/genesis.block

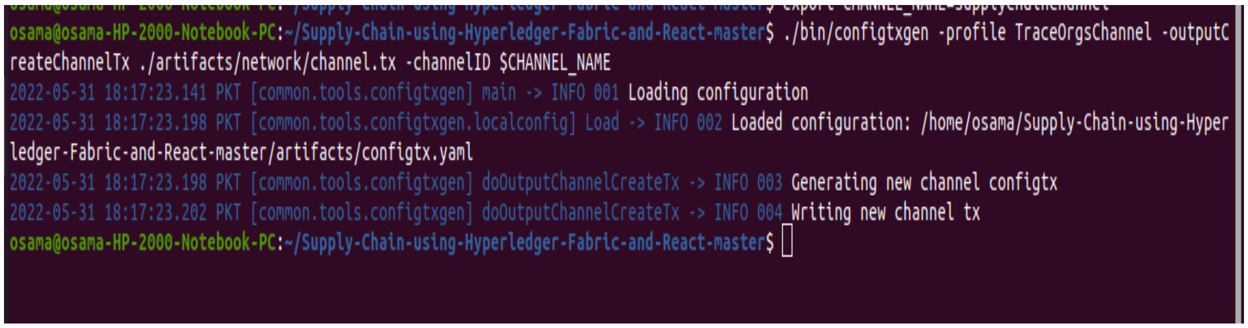


```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/configtxgen -profile TraceOrdererGenesis -outp
utBlock ./artifacts/network/genesis.block -channelID=supply
2022-05-31 18:04:02.694 PKT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2022-05-31 18:04:02.749 PKT [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: solo
2022-05-31 18:04:02.749 PKT [common.tools.configtxgen.localconfig] Load -> INFO 003 Loaded configuration: /home/osama/Supply-Chain-using-Hyper
ledger-Fabric-and-React-master/artifacts/configtx.yaml
2022-05-31 18:04:02.751 PKT [common.tools.configtxgen] doOutputBlock -> INFO 004 Generating genesis block
2022-05-31 18:04:02.752 PKT [common.tools.configtxgen] doOutputBlock -> INFO 005 Writing genesis block
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ 
```

Fig 2

4. export channel name
export CHANNEL_NAME=supplychainchannel

5. generate the channel transaction file
../bin/configtxgen -profile TraceOrgsChannel -outputCreateChannelTx ./artifacts/network/channel.tx -channelID $CHANNEL_NAME

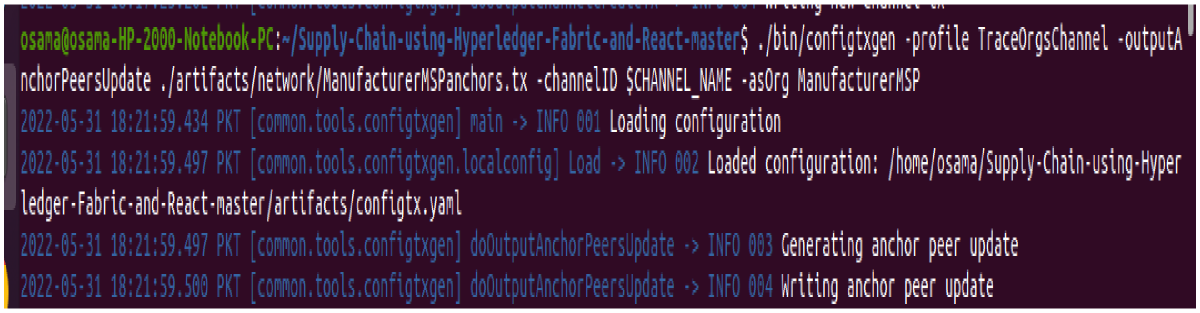


```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/configtxgen -profile TraceOrgsChannel -outputC
reateChannelTx ./artifacts/network/channel.tx -channelID $CHANNEL_NAME
2022-05-31 18:17:23.141 PKT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2022-05-31 18:17:23.198 PKT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/osama/Supply-Chain-using-Hyper
ledger-Fabric-and-React-master/artifacts/configtx.yaml
2022-05-31 18:17:23.198 PKT [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx
2022-05-31 18:17:23.202 PKT [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ 
```

Fig 3

6. update anchor peers

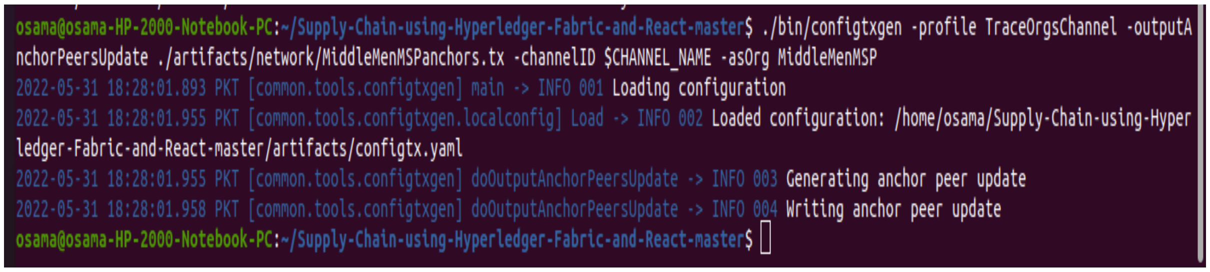
../bin/configtxgen -profile TraceOrgsChannel -outputAnchorPeersUpdate ./artifacts/network/ManufacturerMSPanchors.tx -channelID $CHANNEL_NAME -asOrg ManufacturerMSP



```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/configtxgen -profile TraceOrgsChannel -outputA
nchorPeersUpdate ./artifacts/network/ManufacturerMSPanchors.tx -channelID $CHANNEL_NAME -asOrg ManufacturerMSP
2022-05-31 18:21:59.434 PKT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2022-05-31 18:21:59.497 PKT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/osama/Supply-Chain-using-Hyper
ledger-Fabric-and-React-master/artifacts/configtx.yaml
2022-05-31 18:21:59.497 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2022-05-31 18:21:59.500 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
```

Fig 4

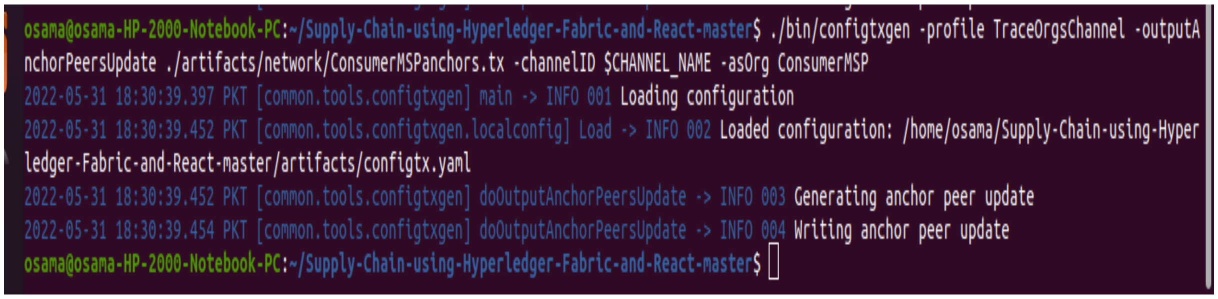
../bin/configtxgen -profile TraceOrgsChannel -outputAnchorPeersUpdate ./artifacts/network/MiddleMenMSPanchors.tx -channelID $CHANNEL_NAME -asOrg MiddleMenMSP



```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/configtxgen -profile TraceOrgsChannel -outputA
nchorPeersUpdate ./artifacts/network/MiddleMenMSPanchors.tx -channelID $CHANNEL_NAME -asOrg MiddleMenMSP
2022-05-31 18:28:01.893 PKT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2022-05-31 18:28:01.955 PKT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/osama/Supply-Chain-using-Hyper
ledger-Fabric-and-React-master/artifacts/configtx.yaml
2022-05-31 18:28:01.955 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2022-05-31 18:28:01.958 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ 
```

Fig 5

../bin/configtxgen -profile TraceOrgsChannel -outputAnchorPeersUpdate ./artifacts/network/ConsumerMSPanchors.tx -channelID $CHANNEL_NAME -asOrg ConsumerMSP



```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ ./bin/configtxgen -profile TraceOrgsChannel -outputA
nchorPeersUpdate ./artifacts/network/ConsumerMSPanchors.tx -channelID $CHANNEL_NAME -asOrg ConsumerMSP
2022-05-31 18:30:39.397 PKT [common.tools.configtxgen] main -> INFO 001 Loading configuration
2022-05-31 18:30:39.452 PKT [common.tools.configtxgen.localconfig] Load -> INFO 002 Loaded configuration: /home/osama/Supply-Chain-using-Hyper
ledger-Fabric-and-React-master/artifacts/configtx.yaml
2022-05-31 18:30:39.452 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update
2022-05-31 18:30:39.454 PKT [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$
```

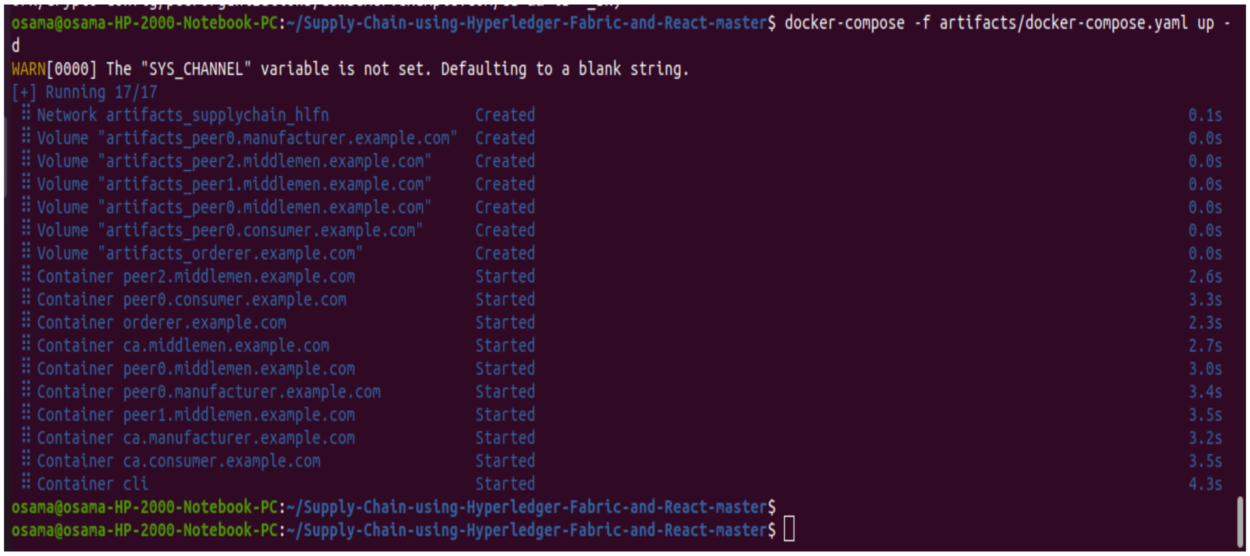
Fig 6

7. up the network
export MANUFACTURER_CA_PRIVATE_KEY=$(cd ./artifacts/network/crypto-config/peerOrganizations/manufacturer.example.com/ca && ls *_sk)

export MIDDLEMEN_CA_PRIVATE_KEY=$(cd ./artifacts/network/crypto-config/peerOrganizations/middlemen.example.com/ca && ls *_sk)

export CONSUMER_CA_PRIVATE_KEY=$(cd ./artifacts/network/crypto-config/peerOrganizations/consumer.example.com/ca && ls *_sk)

docker-compose -f artifacts/docker-compose.yaml up –d



```
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$ docker-compose -f artifacts/docker-compose.yaml up -
d
WARN[0000] The "SYS_CHANNEL" variable is not set. Defaulting to a blank string.
[+] Running 17/17
 ⠿ Network artifacts_supplychain_hlfn                    Created                                                          0.1s
 ⠿ Volume "artifacts_peer0.manufacturer.example.com"     Created                                                          0.0s
 ⠿ Volume "artifacts_peer2.middlemen.example.com"        Created                                                          0.0s
 ⠿ Volume "artifacts_peer1.middlemen.example.com"        Created                                                          0.0s
 ⠿ Volume "artifacts_peer0.middlemen.example.com"        Created                                                          0.0s
 ⠿ Volume "artifacts_peer0.consumer.example.com"         Created                                                          0.0s
 ⠿ Volume "artifacts_orderer.example.com"                Created                                                          0.0s
 ⠿ Container peer2.middlemen.example.com                 Started                                                          2.6s
 ⠿ Container peer0.consumer.example.com                  Started                                                          3.3s
 ⠿ Container orderer.example.com                         Started                                                          2.3s
 ⠿ Container ca.middlemen.example.com                    Started                                                          2.7s
 ⠿ Container peer0.middlemen.example.com                 Started                                                          3.0s
 ⠿ Container peer0.manufacturer.example.com              Started                                                          3.4s
 ⠿ Container peer1.middlemen.example.com                 Started                                                          3.5s
 ⠿ Container ca.manufacturer.example.com                 Started                                                          3.2s
 ⠿ Container ca.consumer.example.com                     Started                                                          3.5s
 ⠿ Container cli                                         Started                                                          4.3s
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$
osama@osama-HP-2000-Notebook-PC:~/Supply-Chain-using-Hyperledger-Fabric-and-React-master$
```

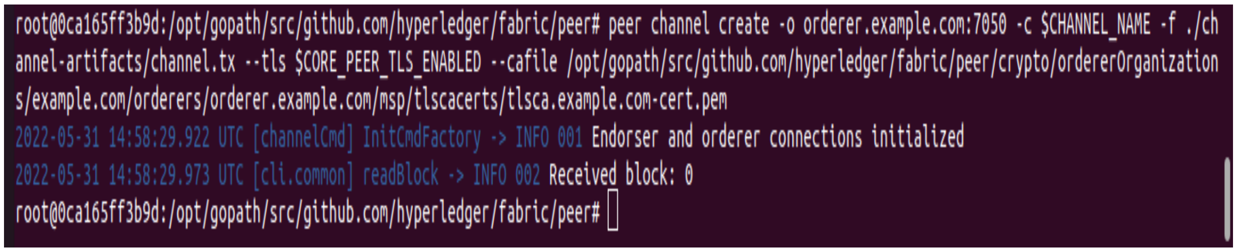
Fig 7

8. get into CLI
docker exec -it cli bash

9. export channel name
export CHANNEL_NAME=supplychainchannel

10. create channel
peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls $CORE_PEER_TLS_ENABLED --cafile

/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

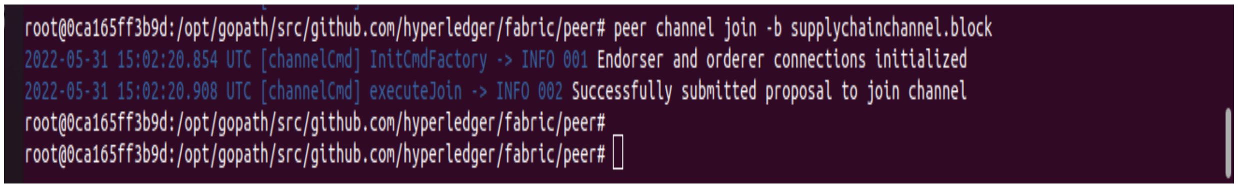


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls $CORE_PEER_TLS_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
2022-05-31 14:58:29.922 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2022-05-31 14:58:29.973 UTC [cli.common] readBlock -> INFO 002 Received block: 0
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 8

11. peer 0 manufacturer join channel

peer channel join -b supplychainchannel.block

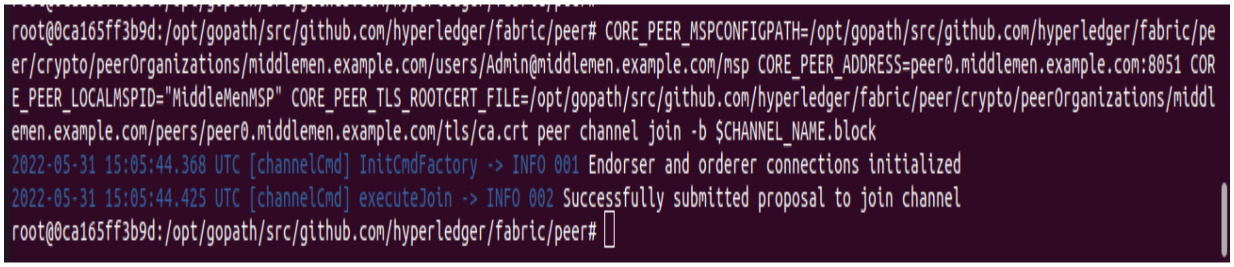


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer channel join -b supplychainchannel.block
2022-05-31 15:02:20.854 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2022-05-31 15:02:20.908 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 9

12. peer 0 middlemen join channel

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/users/Admin@middlemen.example.com/msp
CORE_PEER_ADDRESS=peer0.middlemen.example.com:8051
CORE_PEER_LOCALMSPID="MiddleMenMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/peers/peer0.middlemen.example.com/tls/ca.crt
peer channel join -b $CHANNEL_NAME.block



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/users/Admin@middlemen.example.com/msp CORE_PEER_ADDRESS=peer0.middlemen.example.com:8051 CORE_PEER_LOCALMSPID="MiddleMenMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/peers/peer0.middlemen.example.com/tls/ca.crt peer channel join -b $CHANNEL_NAME.block
2022-05-31 15:05:44.368 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2022-05-31 15:05:44.425 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 10

peer 1 middlemen join channel

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/users/Admin@middlemen.example.com/msp
CORE_PEER_ADDRESS=peer1.middlemen.example.com:9051
CORE_PEER_LOCALMSPID="MiddleMenMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/cryp

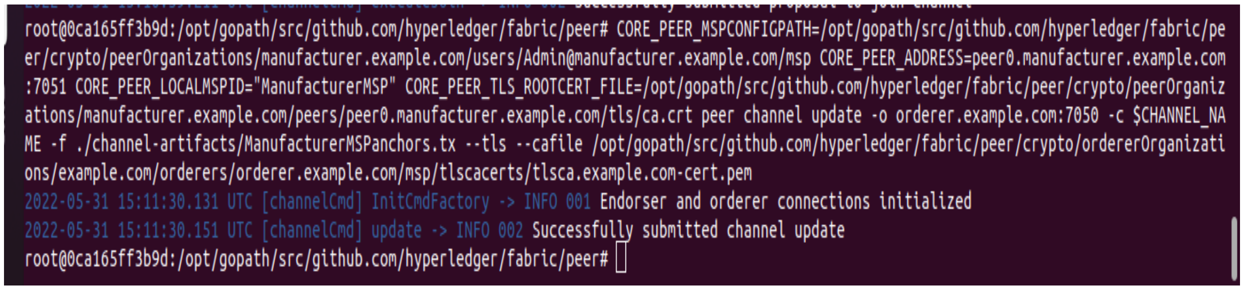
to/peerOrganizations/middlemen.example.com/peers/peer1.middlemen.example.com/tls/ca.crt
peer channel join -b $CHANNEL_NAME.block

peer 2 middlemen join channel
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/users/Admin@middlemen.example.com/msp
CORE_PEER_ADDRESS=peer2.middlemen.example.com:10051
CORE_PEER_LOCALMSPID="MiddleMenMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/peers/peer2.middlemen.example.com/tls/ca.crt
peer channel join -b $CHANNEL_NAME.block

peer 0 consumer join channel
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/consumer.example.com/users/Admin@consumer.example.com/msp
CORE_PEER_ADDRESS=peer0.consumer.example.com:11051
CORE_PEER_LOCALMSPID="ConsumerMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/consumer.example.com/peers/peer0.consumer.example.com/tls/ca.crt peer channel join -b $CHANNEL_NAME.block

13. update anchor peers
peer 0 manufacturer

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/manufacturer.example.com/users/Admin@manufacturer.example.com/msp
CORE_PEER_ADDRESS=peer0.manufacturer.example.com:7051
CORE_PEER_LOCALMSPID="ManufacturerMSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/manufacturer.example.com/peers/peer0.manufacturer.example.com/tls/ca.crt
peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/ManufacturerMSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/pe
er/crypto/peerOrganizations/manufacturer.example.com/users/Admin@manufacturer.example.com/msp CORE_PEER_ADDRESS=peer0.manufacturer.example.com
:7051 CORE_PEER_LOCALMSPID="ManufacturerMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganiz
ations/manufacturer.example.com/peers/peer0.manufacturer.example.com/tls/ca.crt peer channel update -o orderer.example.com:7050 -c $CHANNEL_NA
ME -f ./channel-artifacts/ManufacturerMSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
2022-05-31 15:11:30.131 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2022-05-31 15:11:30.151 UTC [channelCmd] update -> INFO 002 Successfully submitted channel update
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# 
```

Fig 11

peer 0 middlemen

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/users/Admin@middlemen.example.com/msp CORE_PEER_ADDRESS=peer0.middlemen.example.com:8051 CORE_PEER_LOCALMSPID="MiddleMenMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/middlemen.example.com/peers/peer0.middlemen.example.com/tls/ca.crt peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/MiddleMenMSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

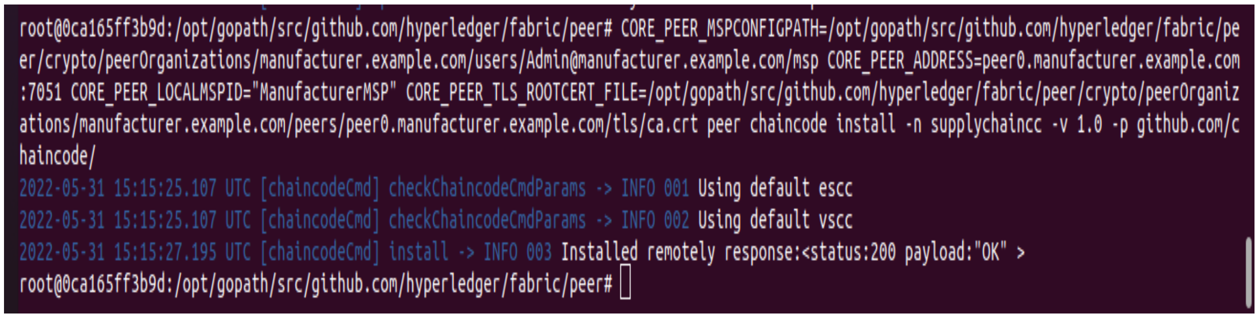
peer 0 consumer

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/consumer.example.com/users/Admin@consumer.example.com/msp CORE_PEER_ADDRESS=peer0.consumer.example.com:11051 CORE_PEER_LOCALMSPID="ConsumerMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/consumer.example.com/peers/peer0.consumer.example.com/tls/ca.crt peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/ConsumerMSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert

14. install chaincode

peer 0 manufacturer

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/manufacturer.example.com/users/Admin@manufacturer.example.com/msp CORE_PEER_ADDRESS=peer0.manufacturer.example.com:7051 CORE_PEER_LOCALMSPID="ManufacturerMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/manufacturer.example.com/peers/peer0.manufacturer.example.com/tls/ca.crt peer chaincode install -n supplychaincc -v 1.0 -p github.com/chaincode/

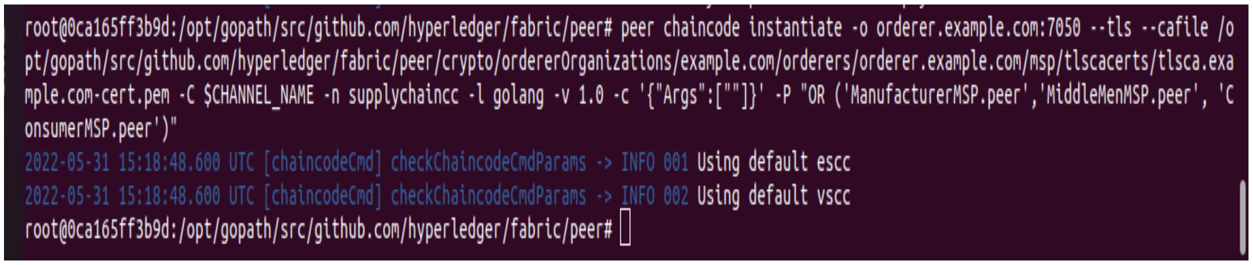


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/pe
er/crypto/peerOrganizations/manufacturer.example.com/users/Admin@manufacturer.example.com/msp CORE_PEER_ADDRESS=peer0.manufacturer.example.com
:7051 CORE_PEER_LOCALMSPID="ManufacturerMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganiz
ations/manufacturer.example.com/peers/peer0.manufacturer.example.com/tls/ca.crt peer chaincode install -n supplychaincc -v 1.0 -p github.com/c
haincode/
2022-05-31 15:15:25.107 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2022-05-31 15:15:25.107 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2022-05-31 15:15:27.195 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 12

15. instantiate or upgrade chaincode

peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -l golang -v 1.0 -c '{"Args":[""]}' -P "OR ('ManufacturerMSP.peer','MiddleMenMSP.peer', 'ConsumerMSP.peer')"



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -l golang -v 1.0 -c '{"Args":[""]}' -P "OR ('ManufacturerMSP.peer','MiddleMenMSP.peer', 'ConsumerMSP.peer')"
2022-05-31 15:18:48.600 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2022-05-31 15:18:48.600 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

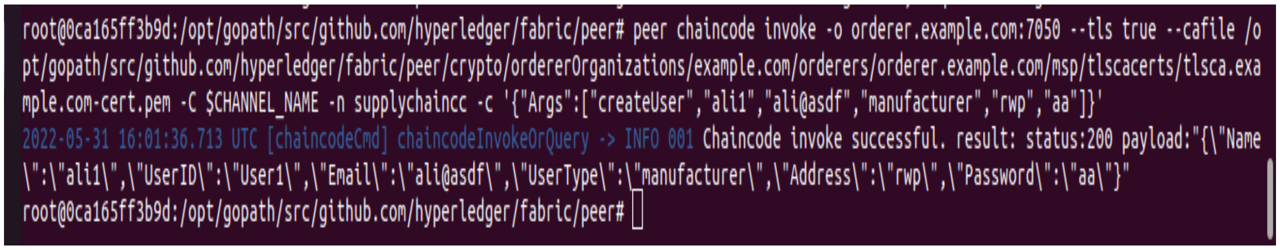
Fig 13

peer chaincode upgrade -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -l golang -v 1.2 -c '{"Args":[""]}' -P "OR ('ManufacturerMSP.peer','MiddleMenMSP.peer', 'ConsumerMSP.peer')"

16. invoke chaincode

create users

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali1","ali@asdf","manufacturer","rwp",”aa”]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali1","ali@asdf","manufacturer","rwp","aa"]}'
2022-05-31 16:01:36.713 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Name\":\"ali1\",\"UserID\":\"User1\",\"Email\":\"ali@asdf\",\"UserType\":\"manufacturer\",\"Address\":\"rwp\",\"Password\":\"aa\"}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 14

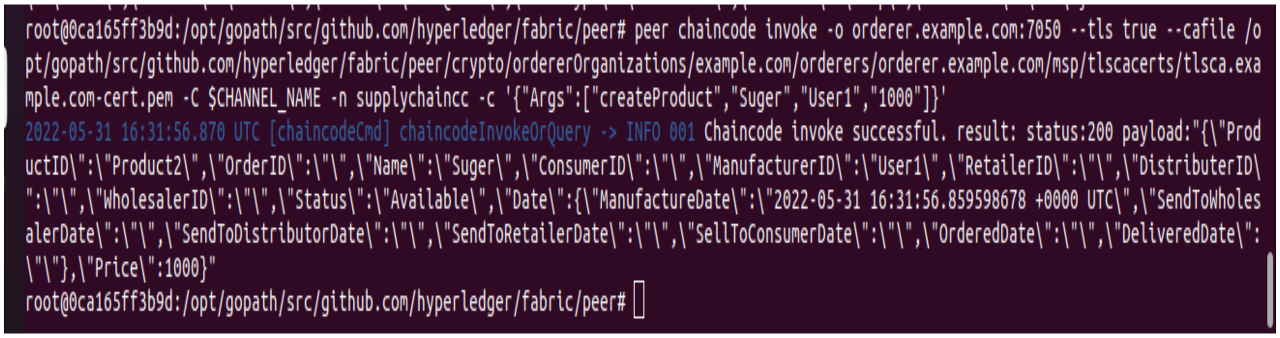
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali2","ali@asdf","wholesaler","rwp",”aa”]}'

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/ord erers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali3","ali@asdf","distributor","rwp",”aa”]}'

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali4","ali@asdf","retailer","rwp",”aa”]}'

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createUser","ali5","ali@asdf","consumer","rwp",”aa”]}'

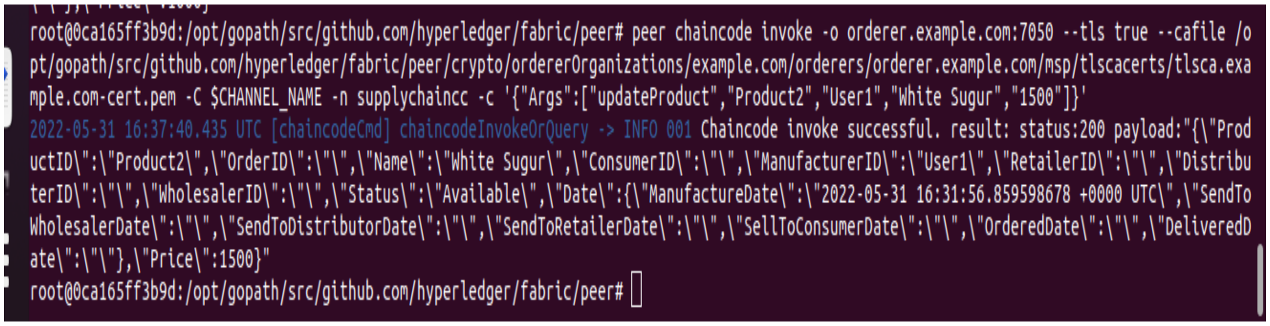
create product
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createProduct","Suger","User1","1000"]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["createProduct","Suger","User1","1000"]}'
2022-05-31 16:31:56.870 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"\",\"Name\":\"Suger\",\"ConsumerID\":\"\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\",\"DistributerID\
":\"\",\"WholesalerID\":\"\",\"Status\":\"Available\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +0000 UTC\",\"SendToWholes
alerDate\":\"\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellToConsumerDate\":\"\",\"OrderedDate\":\"\",\"DeliveredDate\":
\"\"},\"Price\":1000}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 15

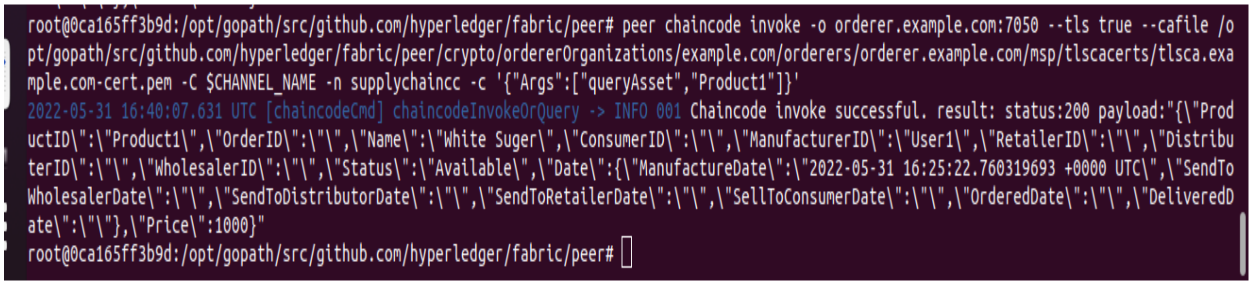
update product
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["updateProduct","Product2","User1","White Sugur","1500"]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["updateProduct","Product2","User1","White Sugur","1500"]}'
2022-05-31 16:37:40.435 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"\",\"Name\":\"White Sugur\",\"ConsumerID\":\"\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\",\"Distribu
terID\":\"\",\"WholesalerID\":\"\",\"Status\":\"Available\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +0000 UTC\",\"SendTo
WholesalerDate\":\"\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellToConsumerDate\":\"\",\"OrderedDate\":\"\",\"DeliveredD
ate\":\"\"},\"Price\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 16

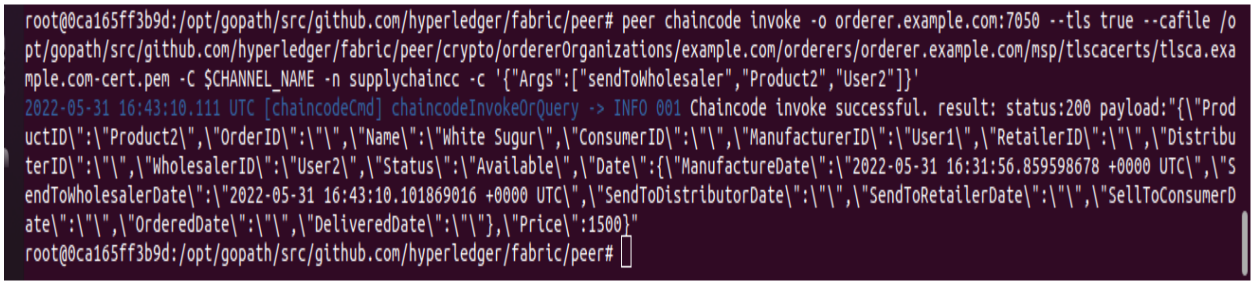
query product
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAsset","Product1"]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAsset","Product1"]}'
2022-05-31 16:40:07.631 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product1\",\"OrderID\":\"\",\"Name\":\"White Suger\",\"ConsumerID\":\"\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\",\"Distribu
terID\":\"\",\"WholesalerID\":\"\",\"Status\":\"Available\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:25:22.760319693 +0000 UTC\",\"SendTo
WholesalerDate\":\"\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellToConsumerDate\":\"\",\"OrderedDate\":\"\",\"DeliveredD
ate\":\"\"},\"Price\":1000}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 17

sending product
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sendToWholesaler","Product2","User2"]}'



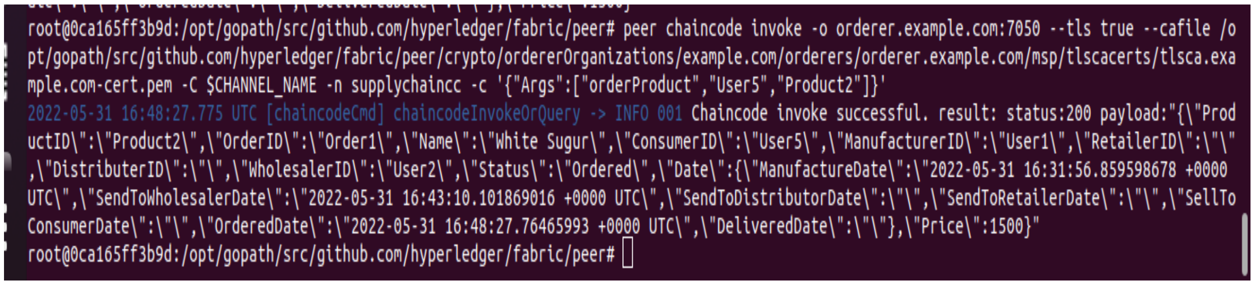
```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sendToWholesaler","Product2","User2"]}'
2022-05-31 16:43:10.111 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"\",\"Name\":\"White Sugur\",\"ConsumerID\":\"\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\",\"Distribu
terID\":\"\",\"WholesalerID\":\"User2\",\"Status\":\"Available\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +0000 UTC\",\"S
endToWholesalerDate\":\"2022-05-31 16:43:10.101869016 +0000 UTC\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellToConsumerD
ate\":\"\",\"OrderedDate\":\"\",\"DeliveredDate\":\"\"},\"Price\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 18

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sendToDistributer","Product2","User3"]}'

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sendToRetailer","Product2","User4"]}'

order product
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["orderProduct","User5","Product2"]}'

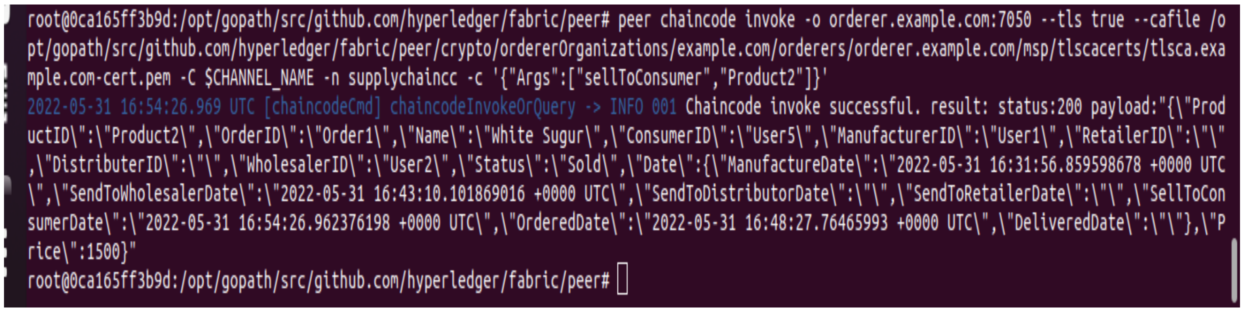


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["orderProduct","User5","Product2"]}'
2022-05-31 16:48:27.775 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"Order1\",\"Name\":\"White Sugur\",\"ConsumerID\":\"User5\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\"
,\"DistributerID\":\"\",\"WholesalerID\":\"User2\",\"Status\":\"Ordered\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +0000
UTC\",\"SendToWholesalerDate\":\"2022-05-31 16:43:10.101869016 +0000 UTC\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellTo
ConsumerDate\":\"\",\"OrderedDate\":\"2022-05-31 16:48:27.76465993 +0000 UTC\",\"DeliveredDate\":\"\"},\"Price\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# 
```

Fig 19

sell to consumer product

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sellToConsumer","Product2"]}'

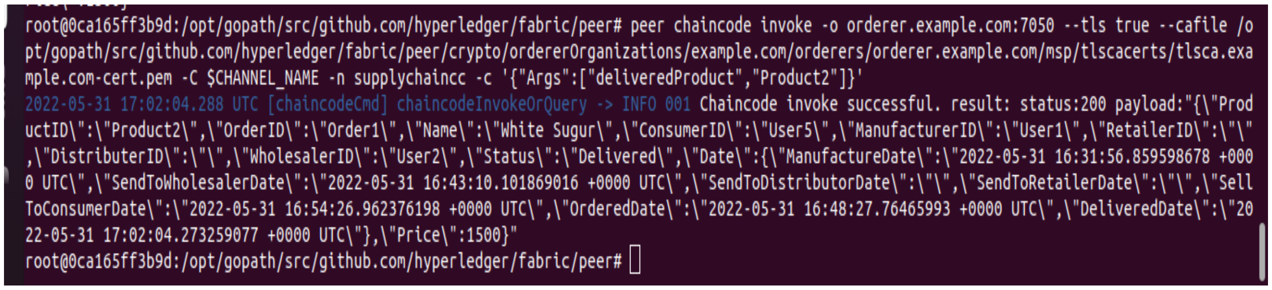


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["sellToConsumer","Product2"]}'
2022-05-31 16:54:26.969 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"Order1\",\"Name\":\"White Sugur\",\"ConsumerID\":\"User5\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\"
,\"DistributerID\":\"\",\"WholesalerID\":\"User2\",\"Status\":\"Sold\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +0000 UTC
\",\"SendToWholesalerDate\":\"2022-05-31 16:43:10.101869016 +0000 UTC\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"SellToCon
sumerDate\":\"2022-05-31 16:54:26.962376198 +0000 UTC\",\"OrderedDate\":\"2022-05-31 16:48:27.76465993 +0000 UTC\",\"DeliveredDate\":\"\"},\"P
rice\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# 
```

Fig 20

delivered product

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["deliveredProduct","Product2"]}'

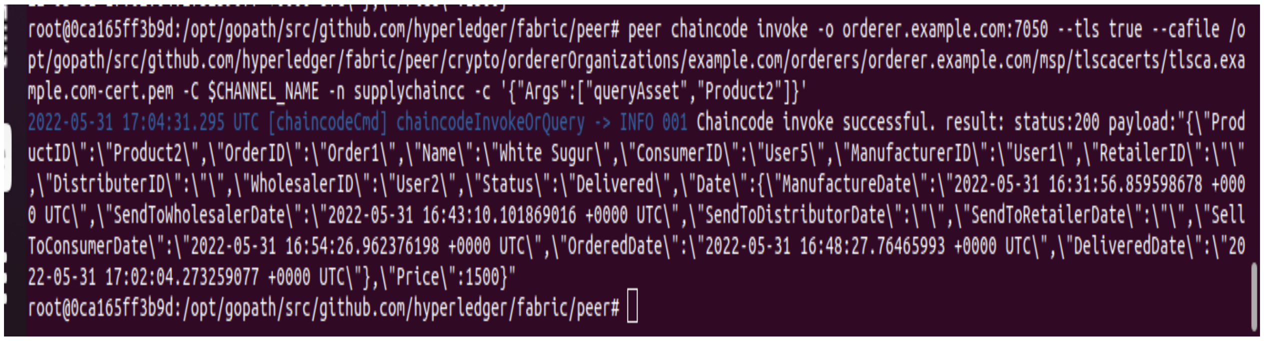


```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["deliveredProduct","Product2"]}'
2022-05-31 17:02:04.288 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"Order1\",\"Name\":\"White Sugur\",\"ConsumerID\":\"User5\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\"
,\"DistributerID\":\"\",\"WholesalerID\":\"User2\",\"Status\":\"Delivered\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +000
0 UTC\",\"SendToWholesalerDate\":\"2022-05-31 16:43:10.101869016 +0000 UTC\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"Sell
ToConsumerDate\":\"2022-05-31 16:54:26.962376198 +0000 UTC\",\"OrderedDate\":\"2022-05-31 16:48:27.76465993 +0000 UTC\",\"DeliveredDate\":\"20
22-05-31 17:02:04.273259077 +0000 UTC\"},\"Price\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# 
```

Fig 21

query asset

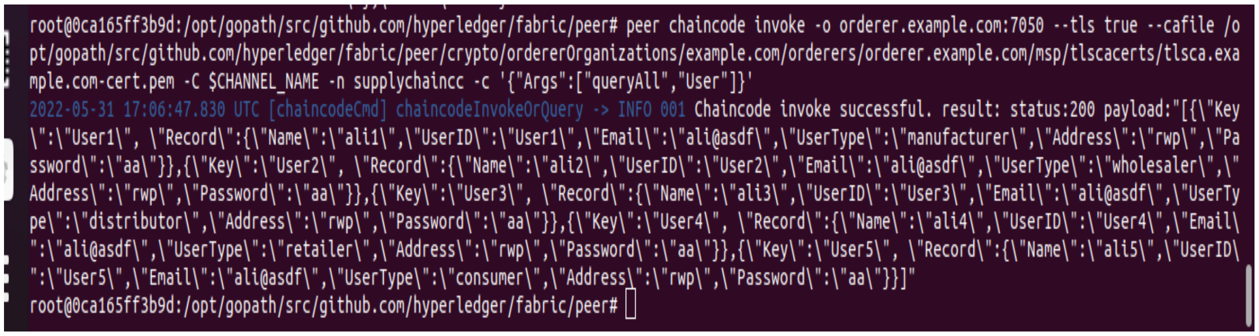
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAsset","Product1"]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAsset","Product2"]}'
2022-05-31 17:04:31.295 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"Prod
uctID\":\"Product2\",\"OrderID\":\"Order1\",\"Name\":\"White Sugur\",\"ConsumerID\":\"User5\",\"ManufacturerID\":\"User1\",\"RetailerID\":\"\"
,\"DistributerID\":\"\",\"WholesalerID\":\"User2\",\"Status\":\"Delivered\",\"Date\":{\"ManufactureDate\":\"2022-05-31 16:31:56.859598678 +000
0 UTC\",\"SendToWholesalerDate\":\"2022-05-31 16:43:10.101869016 +0000 UTC\",\"SendToDistributorDate\":\"\",\"SendToRetailerDate\":\"\",\"Sell
ToConsumerDate\":\"2022-05-31 16:54:26.962376198 +0000 UTC\",\"OrderedDate\":\"2022-05-31 16:48:27.76465993 +0000 UTC\",\"DeliveredDate\":\"20
22-05-31 17:02:04.273259077 +0000 UTC\"},\"Price\":1500}"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 22

query all
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAll","User"]}'



```
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /o
pt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem -C $CHANNEL_NAME -n supplychaincc -c '{"Args":["queryAll","User"]}'
2022-05-31 17:06:47.830 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"[{\"Key
\":\"User1\", \"Record\":{\"Name\":\"ali1\",\"UserID\":\"User1\",\"Email\":\"ali@asdf\",\"UserType\":\"manufacturer\",\"Address\":\"rwp\",\"Pa
ssword\":\"aa\"}},{\"Key\":\"User2\", \"Record\":{\"Name\":\"ali2\",\"UserID\":\"User2\",\"Email\":\"ali@asdf\",\"UserType\":\"wholesaler\",\"
Address\":\"rwp\",\"Password\":\"aa\"}},{\"Key\":\"User3\", \"Record\":{\"Name\":\"ali3\",\"UserID\":\"User3\",\"Email\":\"ali@asdf\",\"UserTy
pe\":\"distributor\",\"Address\":\"rwp\",\"Password\":\"aa\"}},{\"Key\":\"User4\", \"Record\":{\"Name\":\"ali4\",\"UserID\":\"User4\",\"Email\
":\"ali@asdf\",\"UserType\":\"retailer\",\"Address\":\"rwp\",\"Password\":\"aa\"}},{\"Key\":\"User5\", \"Record\":{\"Name\":\"ali5\",\"UserID\
":\"User5\",\"Email\":\"ali@asdf\",\"UserType\":\"consumer\",\"Address\":\"rwp\",\"Password\":\"aa\"}}]"
root@0ca165ff3b9d:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Fig 23

17. down the network
docker-compose -f artifacts/docker-compose.yaml down -v

sudo rm -fR artifacts/network/

docker kill $(docker ps -aq)

docker rm $(docker ps -aq)

docker ps

3.5 REST Server

As the REST Server has a simple and clear design, the implementation process was straightforward as well. The server is implemented with NodeJS [24] was used as Web framework. In order to connect to the Fabric network, fabric-network and fabric-ca-client have been used.

3.6 Front-end application

The front-end application will be built using Node.js. It will have following funtions

- User SignIn
- createProduct
- sendToWholesealer
- sendToDistributor
- sendToRetailer
- sellToCustomer
- Update product
- Email Notification service for above APIs

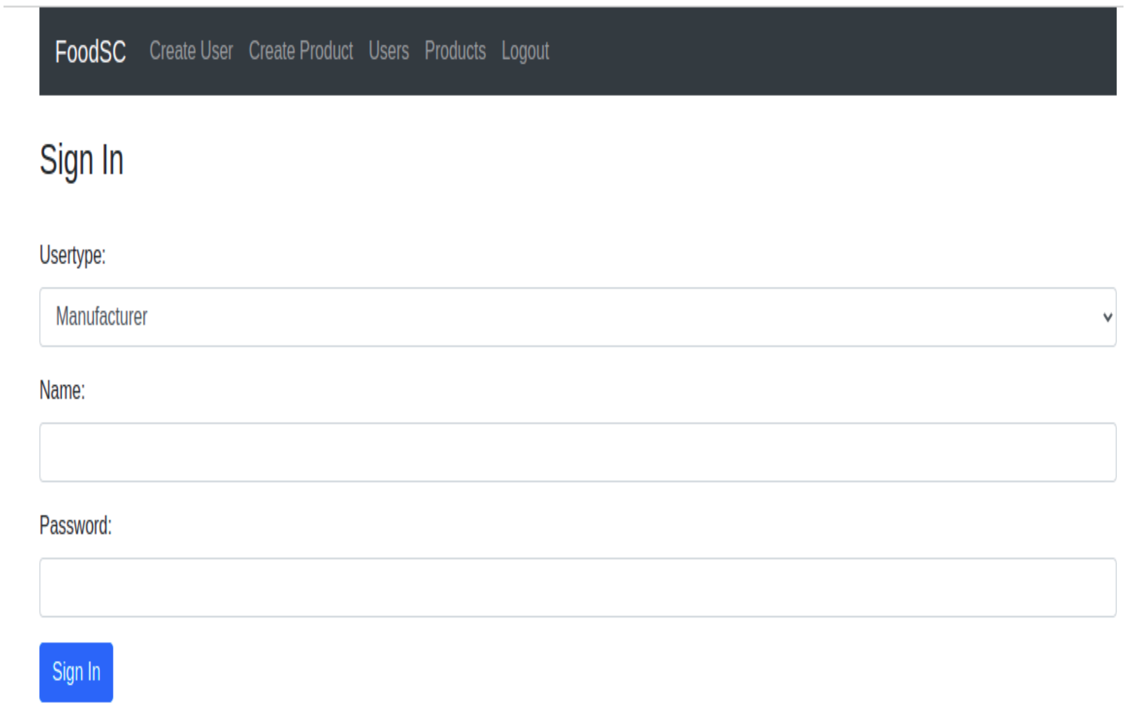
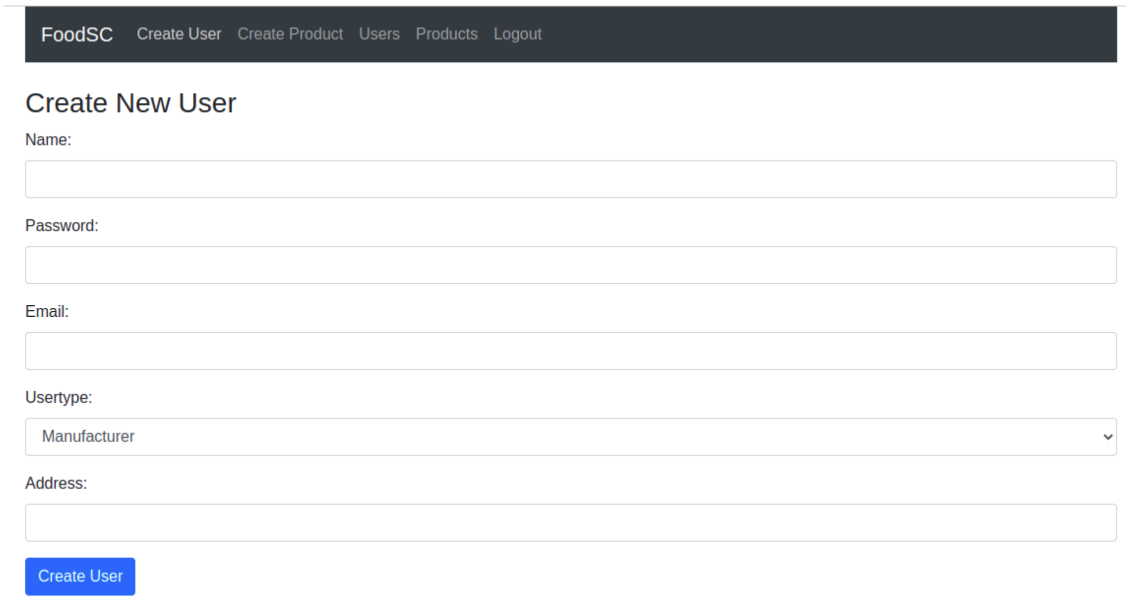


Fig 24



Create New User

Name:

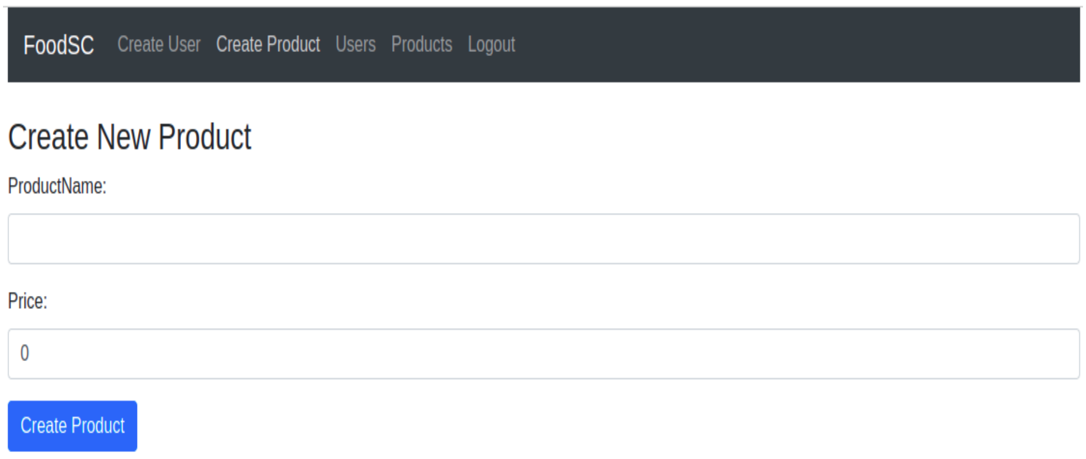
Password:

Email:

Usertype: Manufacturer

Address:

Fig 25

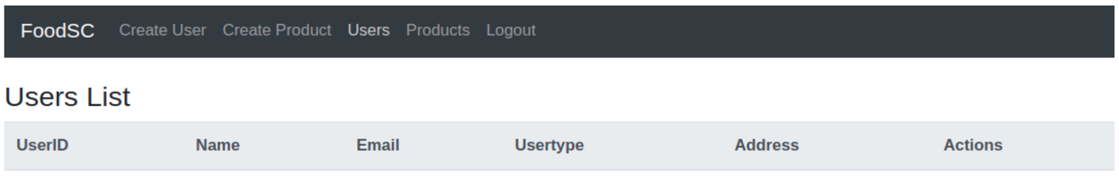


Create New Product

ProductName:

Price: 0

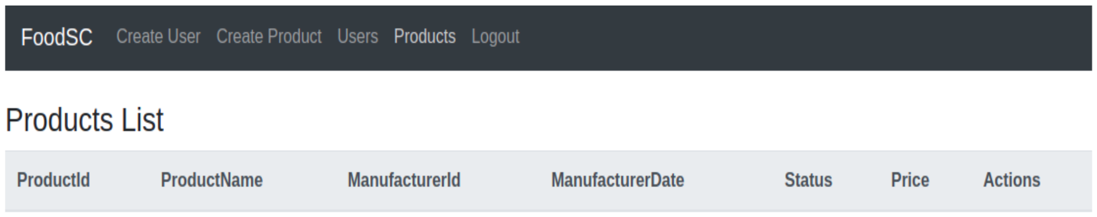
Fig 26



FoodSC Create User Create Product Users Products Logout

Users List

UserID	Name	Email	Usertype	Address	Actions

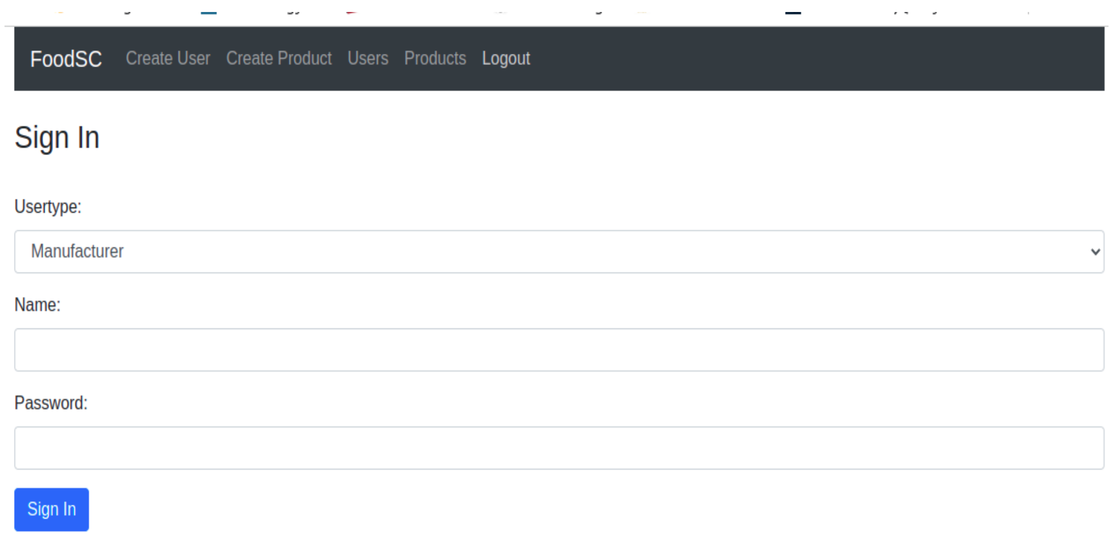


FoodSC Create User Create Product Users Products Logout

Products List

ProductId	ProductName	ManufacturerId	ManufacturerDate	Status	Price	Actions

Fig 28



FoodSC Create User Create Product Users Products Logout

Sign In

Usertype: Manufacturer

Name:

Password:

Sign In

Fig 29

Chapter 4: Conclusion

Distributed ledger technology is a revolutionary innovation with huge potential to improve many current existing systems by making them more transparent, secure and efficient. In this dissertation, the main characteristics of blockchain technology are reviewed, with emphasis on Hyperledger Fabric and the benefits brought by it. More importantly, the focus was put on the applicability of blockchain technology in the supply chain management domain.

Having the proposed system and the main objectives of this paper in mind, we can conclude that blockchain technology can have a great utility in the supply chain management. More specifically, the dispute resolution process in the supply chain can be assisted and improved by making use of the transparency, immutability and security which this technology brings. Among the limitations of this system is the fact that blockchain technology is a relatively immature technology, which can result in resistance or doubt in its adoption process.

Chapter 5: Future Work

Future work might include the development of a more elaborate system, which would take into account the business needs and the topology of a real supply chain. Another direction of research can be the further automation of those decisions in the supply chain, which currently are done manually by its participants, such as settlement negotiation in a dispute.

References and Work Cited

[1] Supply Chain and Logistics Management: Concepts, Methodologies, Tools, and Applications. IGI Global, 2020. isbn: 9781799809456. doi: 10.4018/978-1-7998-0945-6. url: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10. 4018/978-1-7998-0945-6.

[2] S. Chopra and P. Meindl. Supply Chain Management: Strategy, Planning, and Operation. English (US). 6th. Pearson Education, 2016. isbn: 97812920093567.

[3] S. S. Pettersen and B. E. Asbjørnslett. “Assessing the Vulnerability of Supply Chains: Advances from Engineering Systems.” In: Revisiting Supply Chain Risk. Ed. by G. A. Zsidisin

and M. Henke. Cham: Springer International Publishing, 2019, pp. 15–35. isbn: 978-3-030-03813-7. doi: 10.1007/978-3-030-03813-7_2. url: https://doi.org/10.1007/978-3-030-03813-7_2

[4] mhugos, “Four Participants in Every Supply Chain | SCM Globe.”

[5] Diamonds. October 2020. url: https://www.everledger.io/industry-solutions/ diamonds/.

[6] W. Donald. Supply Chain Risk Management : Vulnerability and Resilience in Logistics. Vol. 2nd ed. Kogan Page, 2011, pp. 99–128. isbn: 9780749463939. url: http : //search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=395740& site=ehost-live.

[7] P. Yang, J. Tang, and C. Yan. “A Case Study of Global Supply Chain Risk Management.” In: 2012 24th Chinese Control and Decision Conference (CCDC). 2012, pp. 1996–2000. doi: 10.1109/CCDC.2012.6243026.

[8] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” tech. rep., Manubot, Nov. 2019.

[9] “Blockchain Technology: Beyond Bitcoin (2016), Crosby, Nachiappan, Pattanayak, Verma & Kalyanaraman,” Nov. 2017.

[10] “Ledger — hyperledger-fabricdocs master documentation.” Available at https://hyperledger-fabric. readthedocs.io/en/release-2.0/ledger/ledger.html.

[11] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. BITCOIN, 2008. URL https://bitcoin.org/bitcoin.pdf.

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger

Fabric: A Distributed Operating System for Permissioned Blockchains. pages 1–15, 2018. doi: 10.1145/3190508.3190538. URL http://arxiv.org/abs/1801. 10228. arXiv: 1801.10228.

[13] V. Buterin. On public and private blockchains, 2014. URL https://blog.ethereum.org/ 2015/08/07/on-public-and-private-blockchains/. Accessed: 2021-02-02.

[14] N. Szabo. Formalizing and securing relationships on public networks. First Monday, 2(9), 1997. ISSN 13960466. doi: 10.5210/fm.v2i9.548. URL https://ojphi.org/ojs/ index.php/fm/article/view/548.

[15] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014. URL https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2020-04-28.

[16] O. Kelly, B. Mic, M. James, A. Shawn, M. Dan, and M. Cian. Sawtooth: An introduction, 01 2018. URL https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_ Sawtooth_WhitePaper.pdf. [Online; accessed on 09-May-2021].

[17] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., ´ Cocco, S. W., and Yellick, J. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, EuroSys '18.

[18] Kitti Klinbua and Wiwat Vatanawood. Translating tosca into docker-compose yaml file using antlr. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pages 145–148. IEEE, 2017.

[19] Open Source Blockchain Technologies. October 2020. url: https://www.hyperledger. Org/.

[20] url: https://hub.docker.com/u/hyperledger/.

[21] Node.js. About Node.js. url: https://nodejs.org/en/about/

[22] “Using the Fabric test network — hyperledger-fabricdocs master documentation.”

[23] “hyperledger/fabric-samples,” Oct. 2020. original-date: 2017-06-20T00:15:53Z.

[24] “Express - Node.js web application framework.”