SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

CSE 462 – ADVANCED BLOCK CHAIN TECHNOLOGIES

CA 4 - ASSIGNMENT REPORT

IMPLEMENTATION OF HYPERLEDGER FABRIC AND SAWTOOTH

Submitted to

SRI RAMACHANDRA INSTITUTE OF HIGHER EDUCATION AND RESEARCH
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

For the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING
(Cyber Security and Internet of Things)

By

SHRIRAM KP (E0219007)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SRET, PORUR, CHENNAI- 600116
JANUARY 2023

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that the Internship report submitted by **SHRIRAM KP** is a record of originalwork done and submitted to **SRI RAMACHANDRA ENGINEERING ANDTECHNOLOGY** during the academic year 2022 in partial fulfillment of the requirements forthe award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ANDENGINEERING (Cyber Security and Internet of Things).

Course Faculty

Prof. N. Priya

Sri Ramachandra Engineering and Technology
Porur, Chennai-600116

Vice-Principal

Dr. M. Prema

Sri Ramachandra Engineering and Technology
Porur, Chennai-600116

Evaluation Date:

ACKNOWLEDGEMENT

We express our sincere gratitude to our Chancellor, Vice-Chancellor and our sincere gratitude to our Provost Dr.V. Raju and our Vice-Principal Prof.M. Prema for their support and for providing the required facilities for carrying out this study.

We wish to thank our faculty, Prof. Priya .N, Assistant Professor, Sri Ramachandra Engineering and Technology for extending help and encouragement throughout the project. Without her continuous guidance and persistent help, this project would not have been a success for us.

We are grateful to Department of Computer Science and Engineering, Sri Ramachandra Engineering and Technology, our beloved parents and friends for extending the support, who helped us to overcome obstacles in the studies.

ABSTRACT

Supply chain operations can be made better by Hyperledger Fabric networks by boosting the traceability and transparency of network interactions. While Hyperledger Sawtooth's user-friendly design provides perfect performance for enterprise usage, a Fabric network allows businesses with access to the ledger to view the same immutable data, enforcing accountability and lowering the danger of counterfeiting. To guarantee the greatest streamlined development experience, we maintained the major network and development layer apart in this system. Throughout the development phase, the basic system is untouched. Because of this, each Hyperledger application has its own use cases, and we'll examine fabric and sawtooth while implementing both apps.

Introduction

Numerous Blockchain frameworks and tools have been added to the Blockchain sector throughout the years as the technology has grown and developed. Hyperledger is one such Blockchain framework.

In 2016, Hyperledger became well-known. It is an open-source collection of tools and initiatives created specifically to speed up the creation of Blockchain applications and systems through improved cooperation between companies and developers using the DLT (Distributed Ledger Technology).

Hyperledger Fabric :

The enterprise-grade distributed ledger system Hyperledger Fabric seeks to offer two essential characteristics for Blockchain use cases: flexibility and adaptability. By utilising plug-and-play components like privacy, consensus, and permissioned services, Fabric's modular design adapts to the variety of industry use cases of Blockchain technology with ease.

Core Features of Hyperledger fabric :

- It has a highly modular, permissioned architecture.
- It features a plug-and-play consensus.
- It has an open smart contract model that imparts the flexibility to implement any desired solution model (account model, UTXO model, etc.).
- It has a low latency of finality/confirmation.
- It has support for EVM and Solidity.
- It supports queryable data (key-based queries and JSON queries).
- It features multi-language smart contract support for languages like Go, Java, and Javascript.
- It offers a flexible approach to data privacy – it performs data isolation via ‘channels,’ data sharing on a need-to-know basis by leveraging private data ‘collections.’
- It features a flexible endorsement model for achieving consensus across required organizations.
- It facilitates continuous operations, including rolling upgrades and asymmetric version sup-port.

The permissioned Blockchain network known as Hyperledger Fabric is created by groups of companies who join forces to form a consortium. The entities that participate in this consortium are referred to as members.

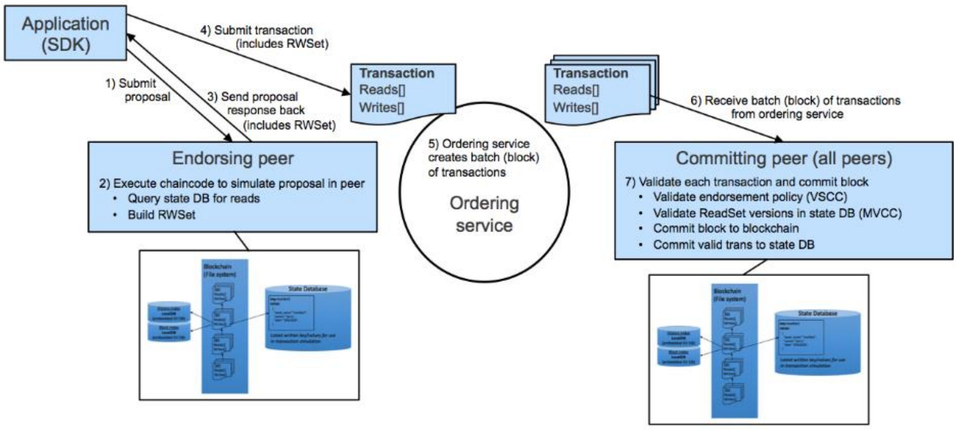
The network-within-network architecture of Fabric is arguably its best feature. Although the network's participants gather with the goal of working together, they keep their internal ties separate because each member organisation needs to protect its own data. Each network participant organisation creates a setup for their fellow network participants. These peers are set up utilising cryptographic components like Certificate Authorities.

The clients inside the company send requests for transaction initiation to these peers within a network. A client in this context refers to any unique application, portal serving a certain corporation, or commercial activity. These clients use the Hyperledger Fabric SDK or REST web API to communicate with the Fabric network. The transaction invocation request is triggered by the chaincode (Smart Contract) that has been installed in the peer nodes.

The Distributed Ledger Technology (DLT) that the Fabric network is based on allows each peer to keep a separate ledger for each channel (that they subscribe to). In contrast to Ethereum, the peers in the Hyperledger Fabric network have distinct functions. There are three categories of peers:

- Endorser peer - Endorser peers are nodes that simulate the transaction's conclusion while validating the transaction and running the chaincode. These peers do not, however, update the ledger.
- Anchor peer - An anchor peer or a group of anchor peers are simultaneously configured during channel configuration. Following their receipt of transaction updates from the endorser peers, these peers broadcast the updates to all other peers within the organisation. Since anchor peers are discoverable, the orderer peer or any other peer can quickly find them.

- Orderer peer: In the Fabric network, orderer peers serve as the main route of communication. The block is created by the orderer peer and sent to all the other peers. It is in charge of ensuring that the ledger state remains constant throughout the network.


Application (SDK)
4) Submit transaction (includes RWSet)
1) Submit proposal
3) Send proposal response back (includes RWSet)
Transaction Reads[] Writes[]
Transaction Reads[] Writes[]
6) Receive batch (block) of transactions from ordering service
Endorsing peer
2) Execute chaincode to simulate proposal in peer
• Query state DB for reads
• Build RWSet
5) Ordering service creates batch (block) of transactions
Ordering service
Committing peer (all peers)
7) Validate each transaction and commit block
• Validate endorsement policy (VSCC)
• Validate ReadSet versions in state DB (MVCC)
• Commit block to blockchain
• Commit valid trans to state DB
Blockchain (File system)
State Database
Blockchain (File system)
State Database

Application of Hyperledger Fabric :

1. Digital payments
2. Food supply chain
3. B2B contracts
4. Digital Identity

Hyperledger Sawtooth :

A distributed ledger technology (blockchain) platform for business is called Hyperledger Sawtooth. Giving fancy names to blockchain platforms is a trend inside a trend, and the blockchain is undoubtedly a long-lasting trend.

The main goal of Hyperledger was to develop a blockchain platform that would be simple for many enterprises and communities to use.

Sawtooth is an operating system for decentralised online communities, data-sharing networks, and microcurrencies. Our design philosophy focuses on making smart contracts secure, especially for enterprise use, and on maintaining the distributed nature of distributed ledgers (essentially, blockchain).

And it appears that Sawtooth's creators were successful in doing so.

Sawtooth Working :

Hyperledger's Ledger

As a result, we can already say that Sawtooth is a decentralised network since it is a distributed ledger platform (shared database). Every member of the network uses a duplicate of the database and follows a procedure to guarantee that everyone agrees on the ledger's current status. Essentially, it is a networking democracy where everyone can participate at any time and make contributions.

HyperLedger Sawtooth consists of three primary components:

1. A data model to record the ledger's present status
2. A language of transactions that users employ to alter the ledger's state
3. A procedure for achieving agreement among the participants.

A transaction family in Sawtooth is a collection of a data model and a transactional language. Although bespoke transaction families are preferred, there are also several pre-built choices available to Hyperledger users.

Algorithm of Proof-elapsed-Time (PoET) Consensus:

Hyperledger Sawtooth decides who can participate (submit transaction) in the distributed ledger platform at the moment and who cannot. In order to make such a decision, blockchain platforms use so-called consensus algorithms. These algorithms come in different shapes. Hyperledger Sawtooth uses proof-of-elapsed time consensus algorithm.

Working Proof-of-Elapsed Time (PoET)

The person in the Sawtooth network who waits the shortest time commits a new block to the ledger. Each member in the Sawtooth network asks a particular amount of randomly selected time. To put it more PoETically, each node sleeps for an unpredictable length of time, and the first one to wake up commits the block and notifies the rest of the network about it so that it can update its state.

We can inquire as to how the distributed ledger platform determines that users do not intentionally implement shorter times in order to win a block. It might compromise the blockchain platform's security.

Additionally, the proof-of-elapsed time technique uses less power than the proof-of-work algorithm, which is employed, for instance, in Bitcoin. Proof-of-elapsed time is a desirable alternative for the business-oriented blockchain platform due to its security and minimal energy usage.

Advantages of Hyperledger Sawtooth

The numerous advantages of Hyperledger Sawtooth make it a powerful and adaptable blockchain technology that is ideal for corporate requirements. These benefits include:

- **Energy efficiency:** As was already established, Sawtooth is more energy-efficient than other blockchain platforms, such as those that employ proof-of-work, due to the characteristics of the proof-of-elapsed time consensus algorithm.
- **Tolerant of Byzantine faults:** To put it another way, the ledger network is guarded against failure that happens when the network is unsure whether or not a certain node has failed.
- **Parallel Planning:** Transaction scheduling is supported in both serial and parallel modes by Hyperledger Sawtooth. By lowering the latency that develops during serial scheduling, parallel scheduling boosts productivity.
- **Support for several languages:** Sawtooth enables the creation of smart contracts in Go, Python, Javascript, Rust, and C++.
- **Dynamic Consensus:** Sawtooth makes it simple to change the consensus algorithm as well as all the blockchain settings without shutting down the entire network.

- **Loose coupling architecture:** Because the blocks in the Sawtooth network are not significantly dependent on one another, changing one of them won't disrupt the network as a whole. Additionally, it makes testing and maintaining the leger much easier and considerably lowers the likelihood of unforeseen problems.
- **Events:** Sawtooth enables the creation and distribution of events over the network.
- **Permissioned/Permissionless:** Sawtooth can be configured to be either permissioned (closed networks; users cannot freely join the distributed ledger system) or permissionless (every user can start interacting with the network, submitting transactions). By default, Hyperledger Sawtooth is permissioned.

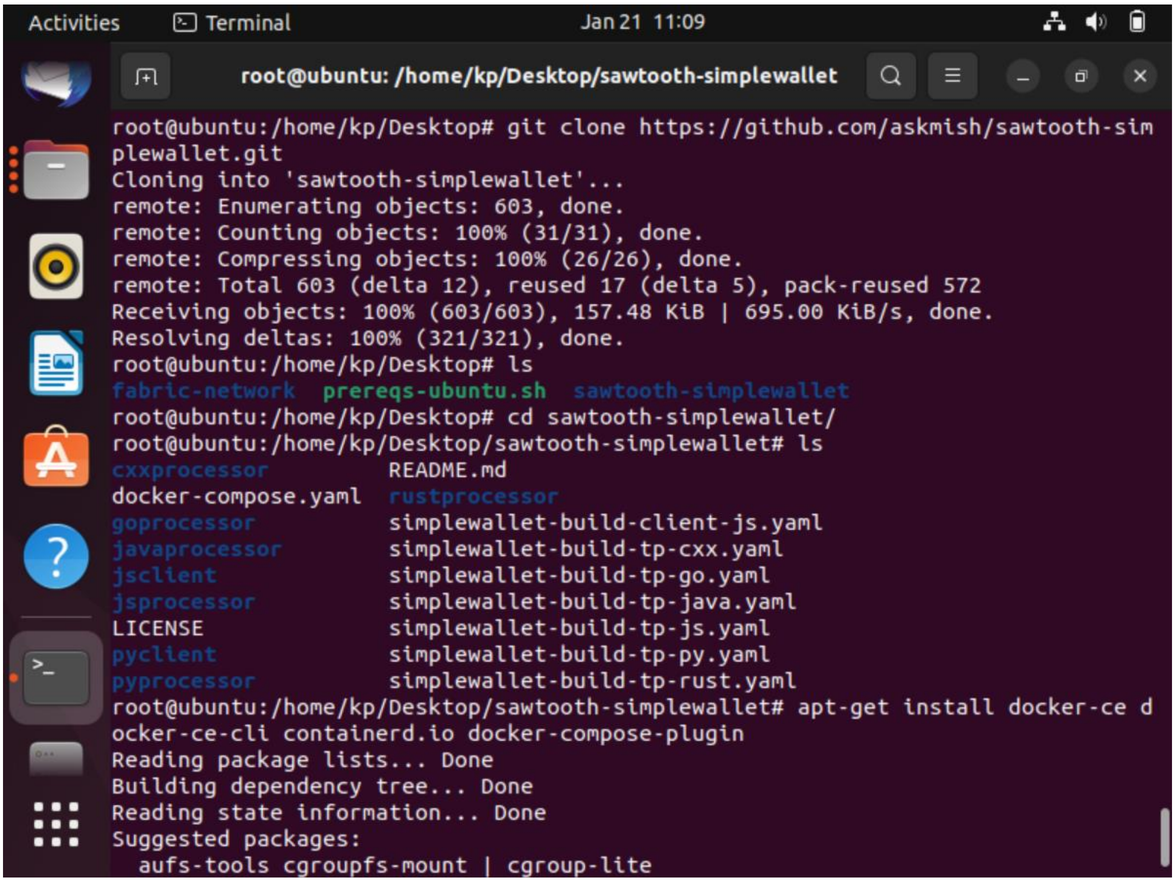
Companies using Hyperledger Sawtooth :

- Aws
- Intel
- IBM
- Huawei
- T-Mobile

Implementation

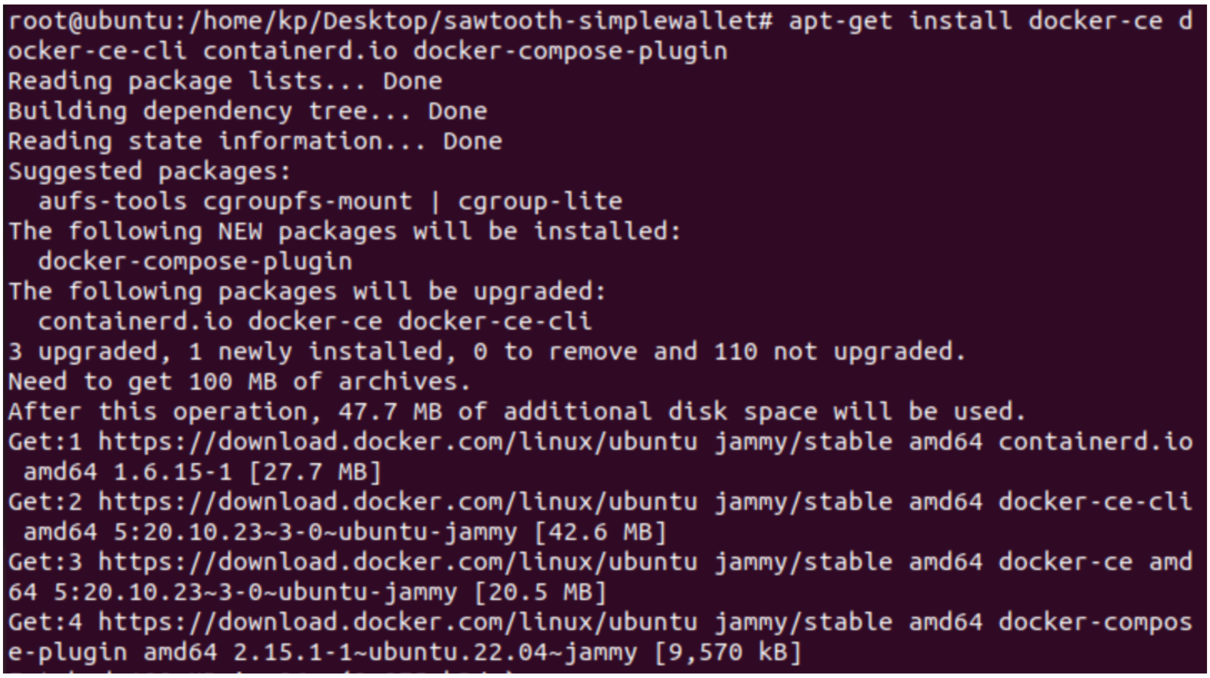
Working of Sawtooth

- Sawtooth simple wallet download



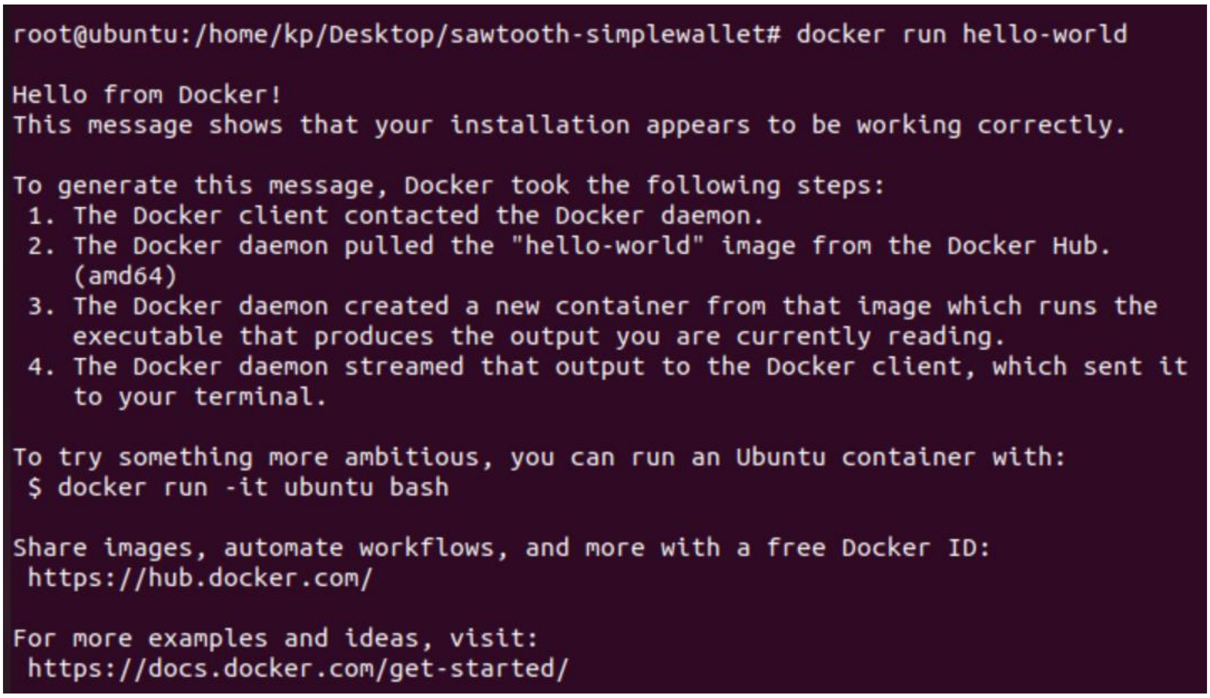
```
root@ubuntu:/home/kp/Desktop# git clone https://github.com/askmish/sawtooth-sim
plewallet.git
Cloning into 'sawtooth-simplewallet'...
remote: Enumerating objects: 603, done.
remote: Counting objects: 100% (31/31), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 603 (delta 12), reused 17 (delta 5), pack-reused 572
Receiving objects: 100% (603/603), 157.48 KiB | 695.00 KiB/s, done.
Resolving deltas: 100% (321/321), done.
root@ubuntu:/home/kp/Desktop# ls
fabric-network  prereqs-ubuntu.sh  sawtooth-simplewallet
root@ubuntu:/home/kp/Desktop# cd sawtooth-simplewallet/
root@ubuntu:/home/kp/Desktop/sawtooth-simplewallet# ls
cxxprocessor         README.md
docker-compose.yaml  rustprocessor
goprocessor          simplewallet-build-client-js.yaml
javaprocessor        simplewallet-build-tp-cxx.yaml
jsclient             simplewallet-build-tp-go.yaml
jsprocessor          simplewallet-build-tp-java.yaml
LICENSE              simplewallet-build-tp-js.yaml
pyclient             simplewallet-build-tp-py.yaml
pyprocessor          simplewallet-build-tp-rust.yaml
root@ubuntu:/home/kp/Desktop/sawtooth-simplewallet# apt-get install docker-ce d
ocker-ce-cli containerd.io docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
```

- Installing docker for using the containers



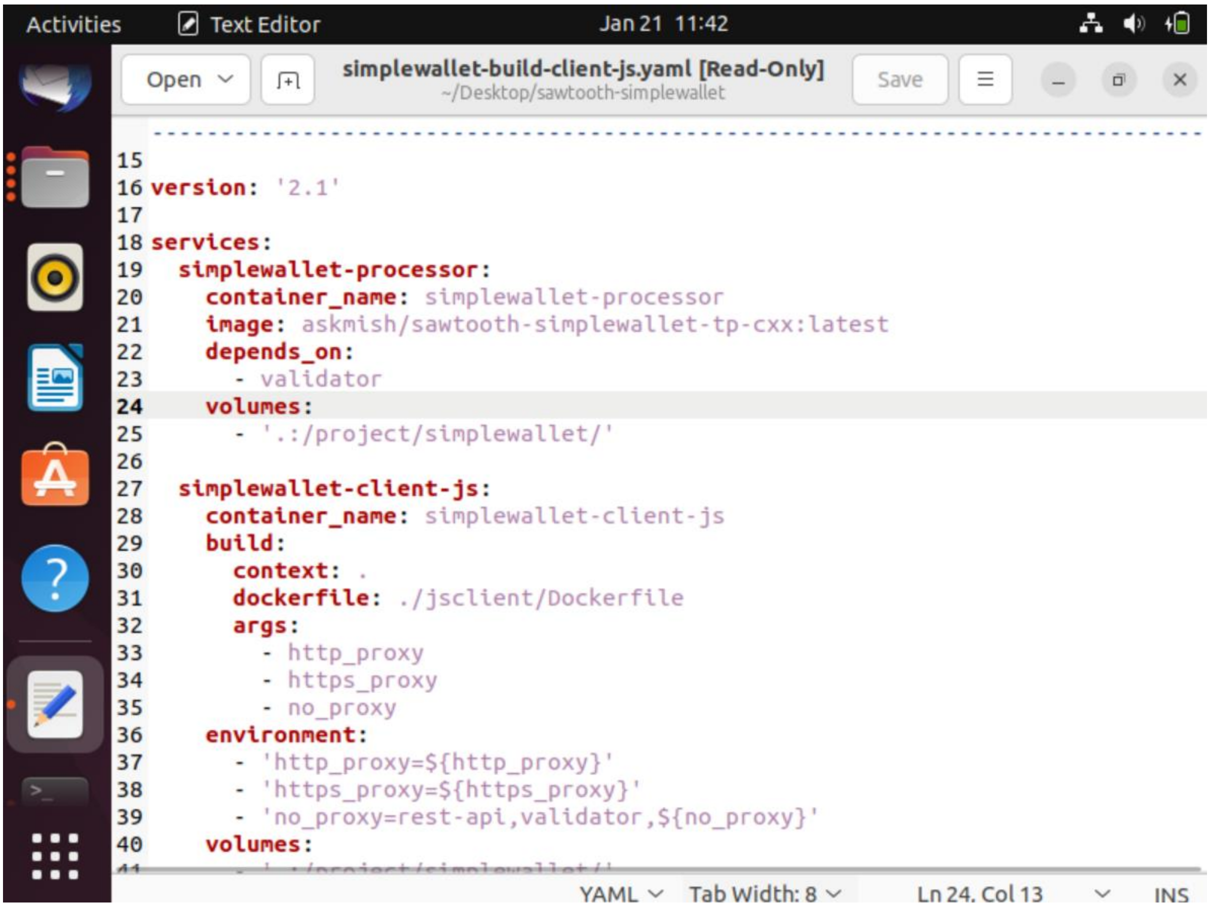
```
root@ubuntu:/home/kp/Desktop/sawtooth-simplewallet# apt-get install docker-ce d
ocker-ce-cli containerd.io docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  docker-compose-plugin
The following packages will be upgraded:
  containerd.io docker-ce docker-ce-cli
3 upgraded, 1 newly installed, 0 to remove and 110 not upgraded.
Need to get 100 MB of archives.
After this operation, 47.7 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io
 amd64 1.6.15-1 [27.7 MB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli
 amd64 5:20.10.23~3-0~ubuntu-jammy [42.6 MB]
Get:3 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd
64 5:20.10.23~3-0~ubuntu-jammy [20.5 MB]
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compos
e-plugin amd64 2.15.1-1~ubuntu.22.04~jammy [9,570 kB]
```

- Checking whether docker is properly installed



```
root@ubuntu:/home/kp/Desktop/sawtooth-simplewallet# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

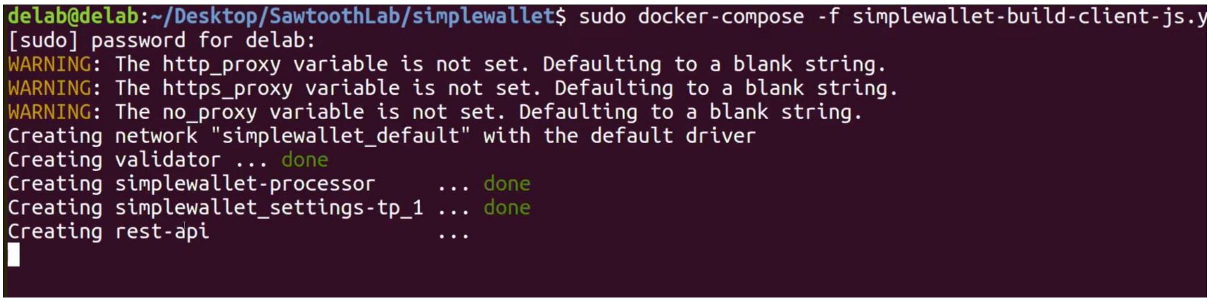
- Build client.yaml file which initiates the application



Activities
Text Editor
Jan 21 11:42
Open
simplewallet-build-client-js.yaml [Read-Only]
~/Desktop/sawtooth-simplewallet
Save
YAML
Tab Width: 8
Ln 24, Col 13
INS

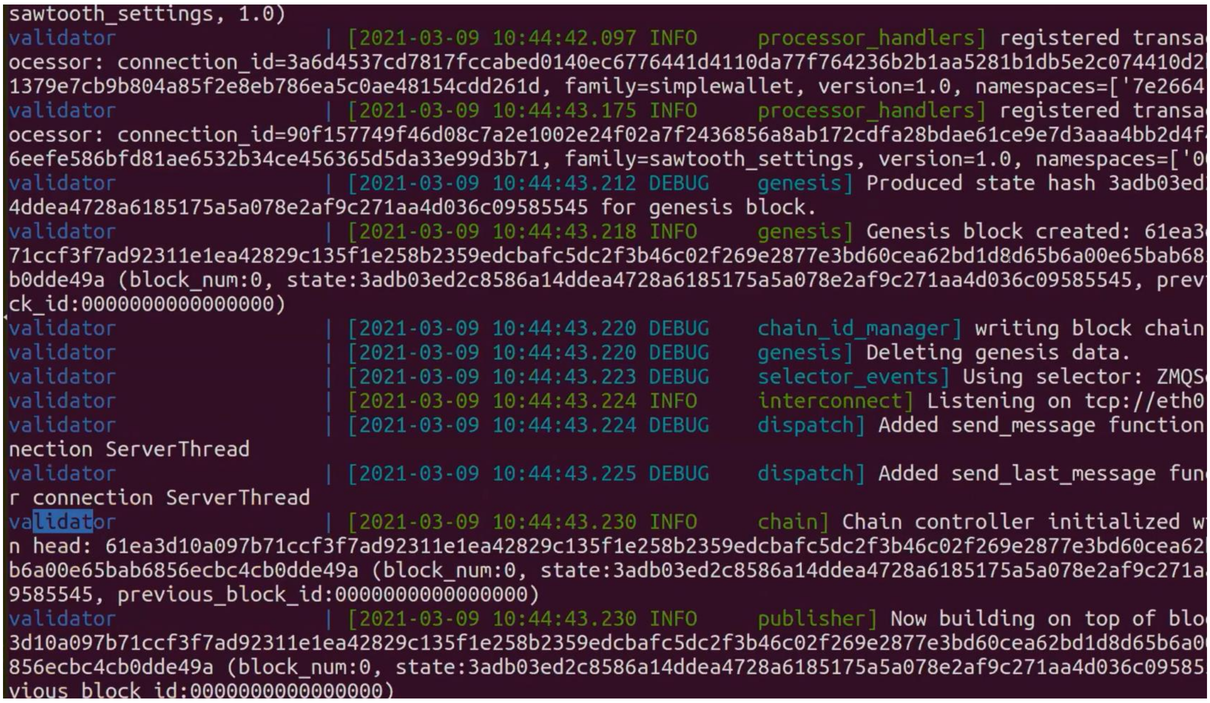
```
---------------------------------------------------------------------------

version: '2.1'

services:
  simplewallet-processor:
    container_name: simplewallet-processor
    image: askmish/sawtooth-simplewallet-tp-cxx:latest
    depends_on:
      - validator
    volumes:
      - '.:/project/simplewallet/'

  simplewallet-client-js:
    container_name: simplewallet-client-js
    build:
      context: .
      dockerfile: ./jsclient/Dockerfile
      args:
        - http_proxy
        - https_proxy
        - no_proxy
    environment:
      - 'http_proxy=${http_proxy}'
      - 'https_proxy=${https_proxy}'
      - 'no_proxy=rest-api,validator,${no_proxy}'
    volumes:
      - '.:/project/simplewallet/'
```

- Uping the server



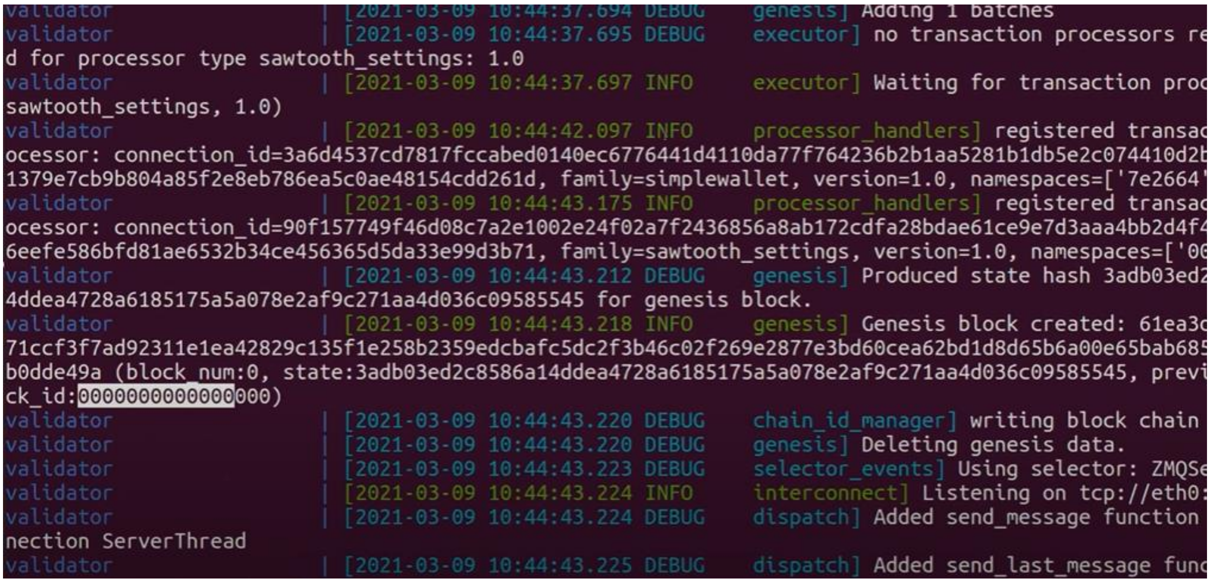
```
delab@delab:~/Desktop/SawtoothLab/simplewallet$ sudo docker-compose -f simplewallet-build-client-js.y
[sudo] password for delab:
WARNING: The http_proxy variable is not set. Defaulting to a blank string.
WARNING: The https_proxy variable is not set. Defaulting to a blank string.
WARNING: The no_proxy variable is not set. Defaulting to a blank string.
Creating network "simplewallet_default" with the default driver
Creating validator ... done
Creating simplewallet-processor     ... done
Creating simplewallet_settings-tp_1 ... done
Creating rest-api                   ...
```

- It creates all the validators and containers



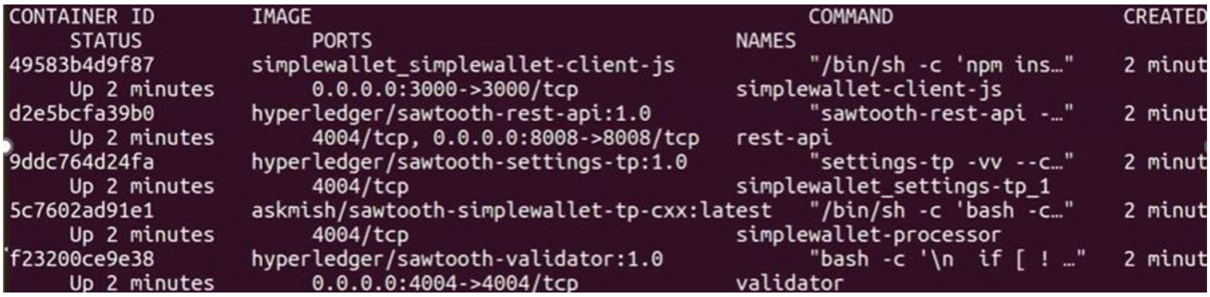
```
sawtooth_settings, 1.0)
validator                    | [2021-03-09 10:44:42.097 INFO     processor_handlers] registered transa
ocessor: connection_id=3a6d4537cd7817fccabed0140ec6776441d4110da77f764236b2b1aa5281b1db5e2c074410d2
1379e7cb9b804a85f2e8eb786ea5c0ae48154cdd261d, family=simplewallet, version=1.0, namespaces=['7e2664
validator                    | [2021-03-09 10:44:43.175 INFO     processor_handlers] registered transa
ocessor: connection_id=90f157749f46d08c7a2e1002e24f02a7f2436856a8ab172cdfa28bdae61ce9e7d3aaa4bb2d4f
6eefe586bfd81ae6532b34ce456365d5da33e99d3b71, family=sawtooth_settings, version=1.0, namespaces=['0
validator                    | [2021-03-09 10:44:43.212 DEBUG    genesis] Produced state hash 3adb03ed
4ddea4728a6185175a5a078e2af9c271aa4d036c09585545 for genesis block.
validator                    | [2021-03-09 10:44:43.218 INFO     genesis] Genesis block created: 61ea3
71ccf3f7ad92311e1ea42829c135f1e258b2359edcbafc5dc2f3b46c02f269e2877e3bd60cea62bd1d8d65b6a00e65bab68
b0dde49a (block_num:0, state:3adb03ed2c8586a14ddea4728a6185175a5a078e2af9c271aa4d036c09585545, prev
ck_id:0000000000000000)
validator                    | [2021-03-09 10:44:43.220 DEBUG    chain_id_manager] writing block chain
validator                    | [2021-03-09 10:44:43.220 DEBUG    genesis] Deleting genesis data.
validator                    | [2021-03-09 10:44:43.223 DEBUG    selector_events] Using selector: ZMQS
validator                    | [2021-03-09 10:44:43.224 INFO     interconnect] Listening on tcp://eth0
validator                    | [2021-03-09 10:44:43.224 DEBUG    dispatch] Added send_message function
nection ServerThread
validator                    | [2021-03-09 10:44:43.225 DEBUG    dispatch] Added send_last_message fun
r connection ServerThread
validator                    | [2021-03-09 10:44:43.230 INFO     chain] Chain controller initialized w
n head: 61ea3d10a097b71ccf3f7ad92311e1ea42829c135f1e258b2359edcbafc5dc2f3b46c02f269e2877e3bd60cea62
b6a00e65bab6856ecbc4cb0dde49a (block_num:0, state:3adb03ed2c8586a14ddea4728a6185175a5a078e2af9c271a
9585545, previous_block_id:0000000000000000)
validator                    | [2021-03-09 10:44:43.230 INFO     publisher] Now building on top of blo
3d10a097b71ccf3f7ad92311e1ea42829c135f1e258b2359edcbafc5dc2f3b46c02f269e2877e3bd60cea62bd1d8d65b6a0
856ecbc4cb0dde49a (block_num:0, state:3adb03ed2c8586a14ddea4728a6185175a5a078e2af9c271aa4d036c09585
vious_block_id:0000000000000000)
```

- Genesis block is created



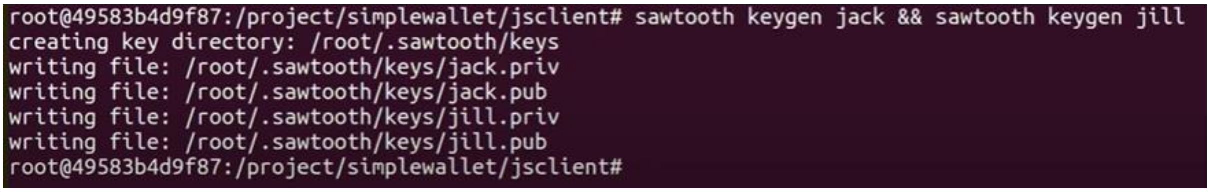
```
validator                    | [2021-03-09 10:44:37.694 DEBUG    genesis] Adding 1 batches
validator                    | [2021-03-09 10:44:37.695 DEBUG    executor] no transaction processors re
d for processor type sawtooth_settings: 1.0
validator                    | [2021-03-09 10:44:37.697 INFO     executor] Waiting for transaction proc
sawtooth_settings, 1.0)
validator                    | [2021-03-09 10:44:42.097 INFO     processor_handlers] registered transac
ocessor: connection_id=3a6d4537cd7817fccabed0140ec6776441d4110da77f764236b2b1aa5281b1db5e2c074410d2b
1379e7cb9b804a85f2e8eb786ea5c0ae48154cdd261d, family=simplewallet, version=1.0, namespaces=['7e2664'
validator                    | [2021-03-09 10:44:43.175 INFO     processor_handlers] registered transac
ocessor: connection_id=90f157749f46d08c7a2e1002e24f02a7f2436856a8ab172cdfa28bdae61ce9e7d3aaa4bb2d4f4
6eefe586bfd81ae6532b34ce456365d5da33e99d3b71, family=sawtooth_settings, version=1.0, namespaces=['00
validator                    | [2021-03-09 10:44:43.212 DEBUG    genesis] Produced state hash 3adb03ed2
4ddea4728a6185175a5a078e2af9c271aa4d036c09585545 for genesis block.
validator                    | [2021-03-09 10:44:43.218 INFO     genesis] Genesis block created: 61ea3c
71ccf3f7ad92311e1ea42829c135f1e258b2359edcbafc5dc2f3b46c02f269e2877e3bd60cea62bd1d8d65b6a00e65bab685
b0dde49a (block_num:0, state:3adb03ed2c8586a14ddea4728a6185175a5a078e2af9c271aa4d036c09585545, previ
ck_id:0000000000000000)
validator                    | [2021-03-09 10:44:43.220 DEBUG    chain_id_manager] writing block chain
validator                    | [2021-03-09 10:44:43.220 DEBUG    genesis] Deleting genesis data.
validator                    | [2021-03-09 10:44:43.223 DEBUG    selector_events] Using selector: ZMQSe
validator                    | [2021-03-09 10:44:43.224 INFO     interconnect] Listening on tcp://eth0:
validator                    | [2021-03-09 10:44:43.224 DEBUG    dispatch] Added send_message function
nection ServerThread
validator                    | [2021-03-09 10:44:43.225 DEBUG    dispatch] Added send_last_message func
```

- Containers running in docker



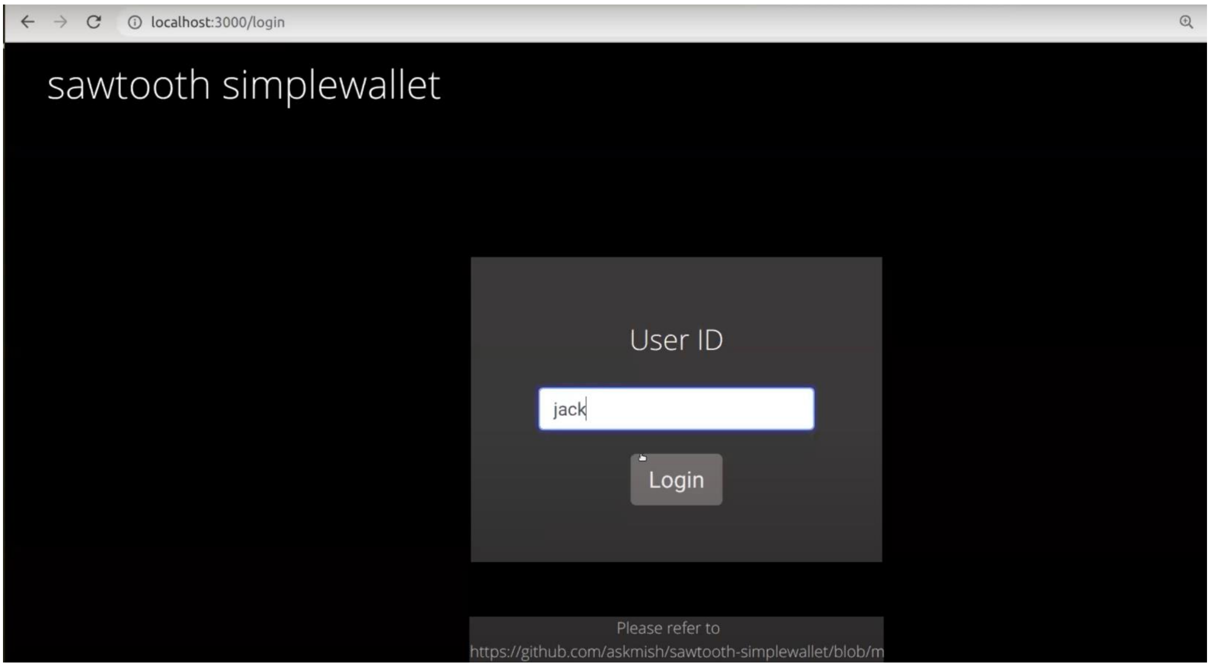
```
CONTAINER ID        IMAGE                                      COMMAND                  CREATED
     STATUS              PORTS                                NAMES
49583b4d9f87        simplewallet_simplewallet-client-js              "/bin/sh -c 'npm ins…"   2 minut
     Up 2 minutes        0.0.0.0:3000->3000/tcp               simplewallet-client-js
d2e5bcfa39b0        hyperledger/sawtooth-rest-api:1.0                "sawtooth-rest-api -…"   2 minut
     Up 2 minutes        4004/tcp, 0.0.0.0:8008->8008/tcp     rest-api
9ddc764d24fa        hyperledger/sawtooth-settings-tp:1.0             "settings-tp -vv --c…"   2 minut
     Up 2 minutes        4004/tcp                             simplewallet_settings-tp_1
5c7602ad91e1        askmish/sawtooth-simplewallet-tp-cxx:latest   "/bin/sh -c 'bash -c…"   2 minut
     Up 2 minutes        4004/tcp                             simplewallet-processor
f23200ce9e38        hyperledger/sawtooth-validator:1.0               "bash -c '\n  if [ ! …"   2 minut
     Up 2 minutes        0.0.0.0:4004->4004/tcp               validator
```

- Creating a public key and private key for two users



```
root@49583b4d9f87:/project/simplewallet/jsclient# sawtooth keygen jack && sawtooth keygen jill
creating key directory: /root/.sawtooth/keys
writing file: /root/.sawtooth/keys/jack.priv
writing file: /root/.sawtooth/keys/jack.pub
writing file: /root/.sawtooth/keys/jill.priv
writing file: /root/.sawtooth/keys/jill.pub
root@49583b4d9f87:/project/simplewallet/jsclient#
```

- As the application is running at localhost we will view the website



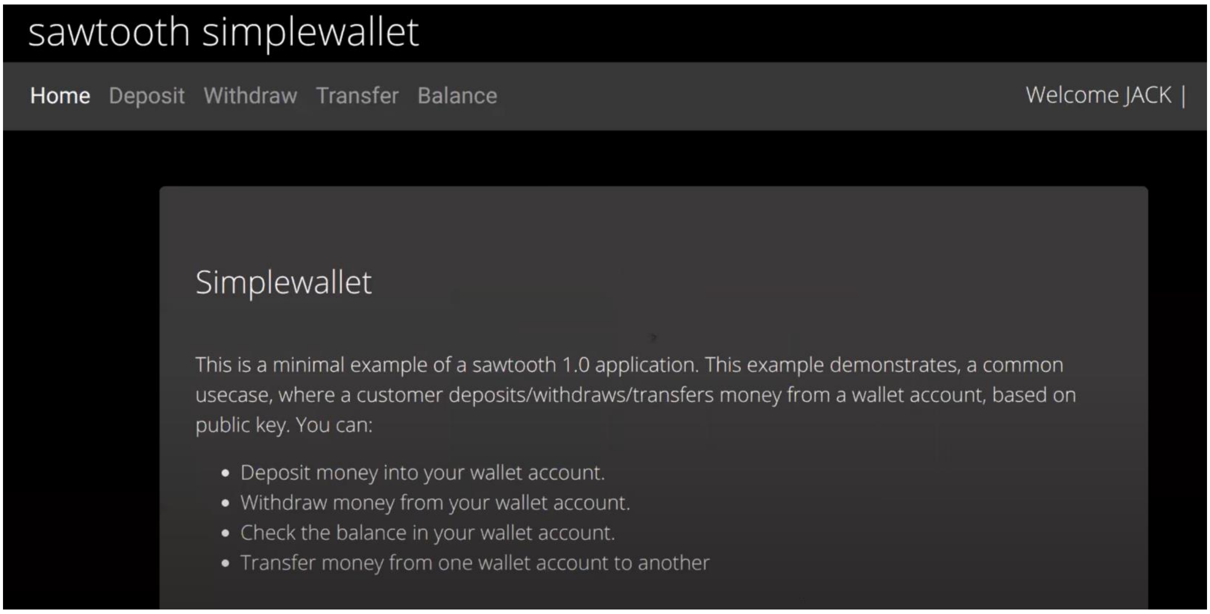
localhost:3000/login

sawtooth simplewallet

User ID: jack

Please refer to
https://github.com/askmish/sawtooth-simplewallet/blob/m

- Jack username has been logged in



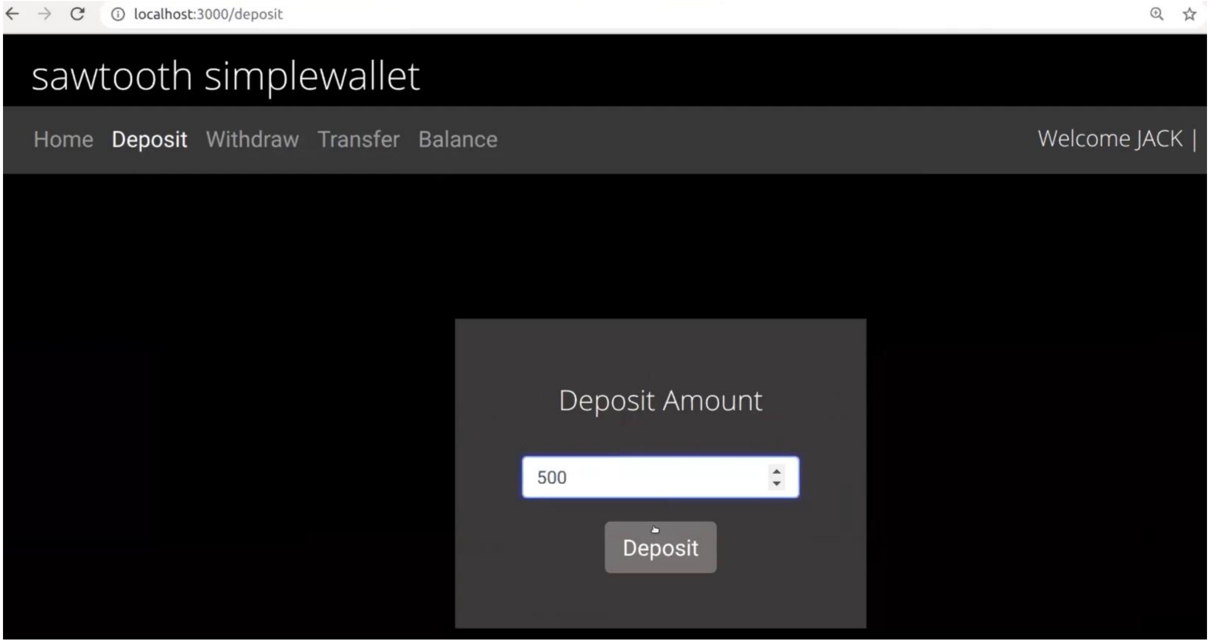
sawtooth simplewallet

Simplewallet

This is a minimal example of a sawtooth 1.0 application. This example demonstrates, a common usecase, where a customer deposits/withdraws/transfers money from a wallet account, based on public key. You can:

- Deposit money into your wallet account.
- Withdraw money from your wallet account.
- Check the balance in your wallet account.
- Transfer money from one wallet account to another

- Lets do some transaction for the blocks to be added



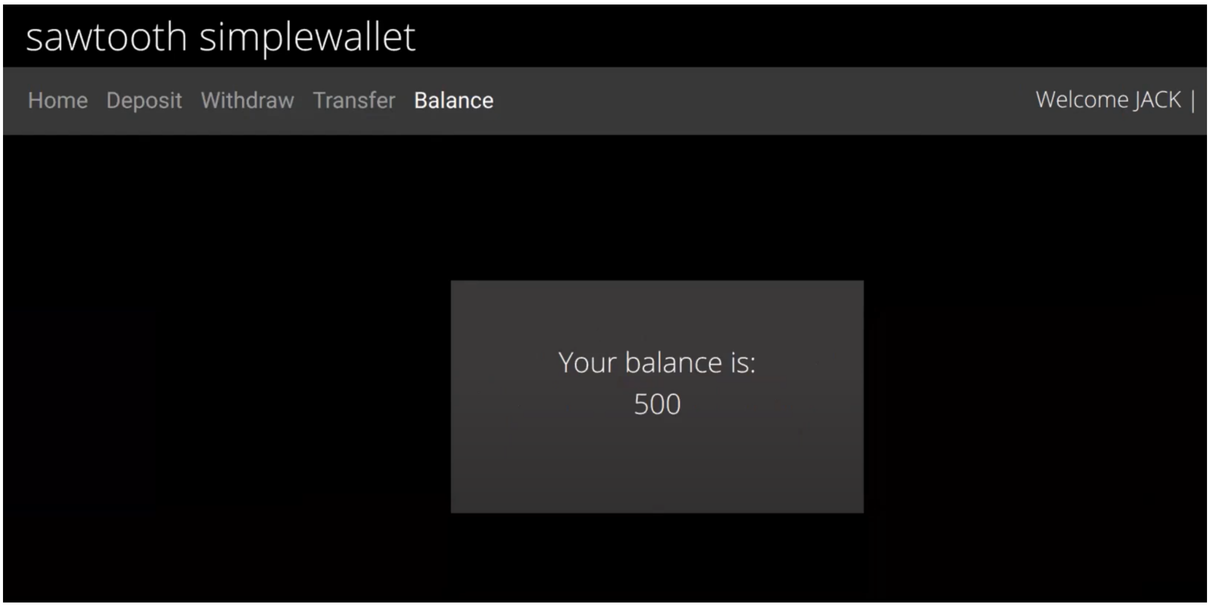
localhost:3000/deposit

sawtooth simplewallet

Deposit Amount

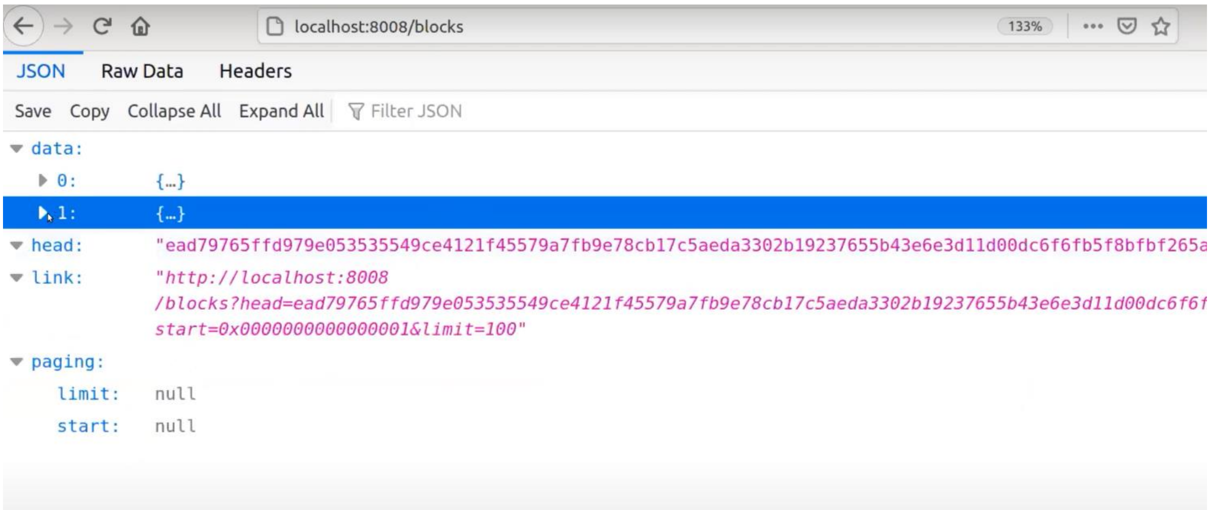
500

- We can see the balance has been updated



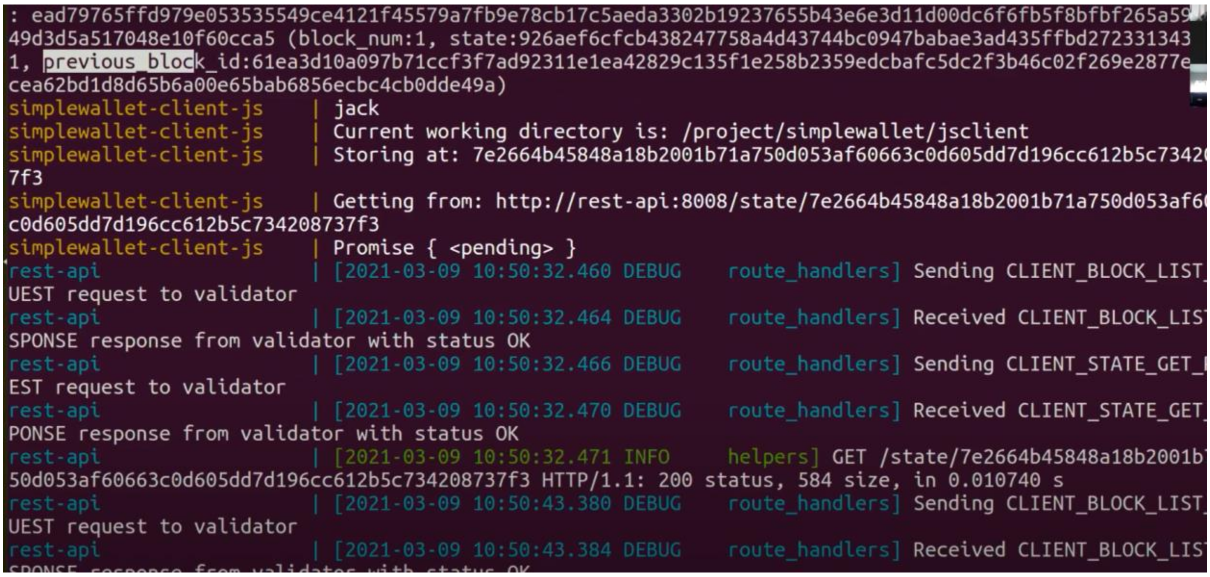
Your balance is:
500

- Two blocks are created



```
▼ data:
    ▶ 0:      {…}
    ▶ 1:      {…}
▼ head:       "ead79765ffd979e053535549ce4121f45579a7fb9e78cb17c5aeda3302b19237655b43e6e3d11d00dc6f6fb5f8bfbf265a
▼ link:       "http://localhost:8008
              /blocks?head=ead79765ffd979e053535549ce4121f45579a7fb9e78cb17c5aeda3302b19237655b43e6e3d11d00dc6f6f
              start=0x0000000000000001&limit=100"
▼ paging:
      limit:  null
      start:  null
```

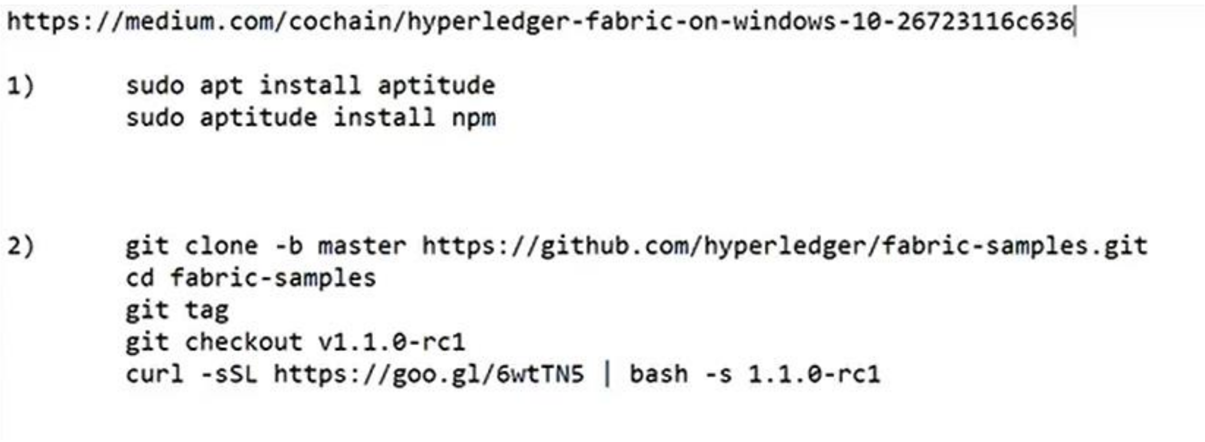
- We can see in the terminal running too update the creation of the block



```
: ead79765ffd979e053535549ce4121f45579a7fb9e78cb17c5aeda3302b19237655b43e6e3d11d00dc6f6fb5f8bfbf265a59
49d3d5a517048e10f60cca5 (block_num:1, state:926aef6cfcb438247758a4d43744bc0947babae3ad435ffbd272331343
1, previous_block_id:61ea3d10a097b71ccf3f7ad92311e1ea42829c135f1e258b2359edcbafc5dc2f3b46c02f269e2877e
cea62bd1d8d65b6a00e65bab6856ecbc4cb0dde49a)
simplewallet-client-js    | jack
simplewallet-client-js    | Current working directory is: /project/simplewallet/jsclient
simplewallet-client-js    | Storing at: 7e2664b45848a18b2001b71a750d053af60663c0d605dd7d196cc612b5c7342
7f3
simplewallet-client-js    | Getting from: http://rest-api:8008/state/7e2664b45848a18b2001b71a750d053af6
c0d605dd7d196cc612b5c734208737f3
simplewallet-client-js    | Promise { <pending> }
rest-api                  | [2021-03-09 10:50:32.460 DEBUG    route_handlers] Sending CLIENT_BLOCK_LIST_
UEST request to validator
rest-api                  | [2021-03-09 10:50:32.464 DEBUG    route_handlers] Received CLIENT_BLOCK_LIS
SPONSE response from validator with status OK
rest-api                  | [2021-03-09 10:50:32.466 DEBUG    route_handlers] Sending CLIENT_STATE_GET_
EST request to validator
rest-api                  | [2021-03-09 10:50:32.470 DEBUG    route_handlers] Received CLIENT_STATE_GET_
PONSE response from validator with status OK
rest-api                  | [2021-03-09 10:50:32.471 INFO     helpers] GET /state/7e2664b45848a18b2001b
50d053af60663c0d605dd7d196cc612b5c734208737f3 HTTP/1.1: 200 status, 584 size, in 0.010740 s
rest-api                  | [2021-03-09 10:50:43.380 DEBUG    route_handlers] Sending CLIENT_BLOCK_LIST_
UEST request to validator
rest-api                  | [2021-03-09 10:50:43.384 DEBUG    route_handlers] Received CLIENT_BLOCK_LIS
SPONSE response from validator with status OK
```

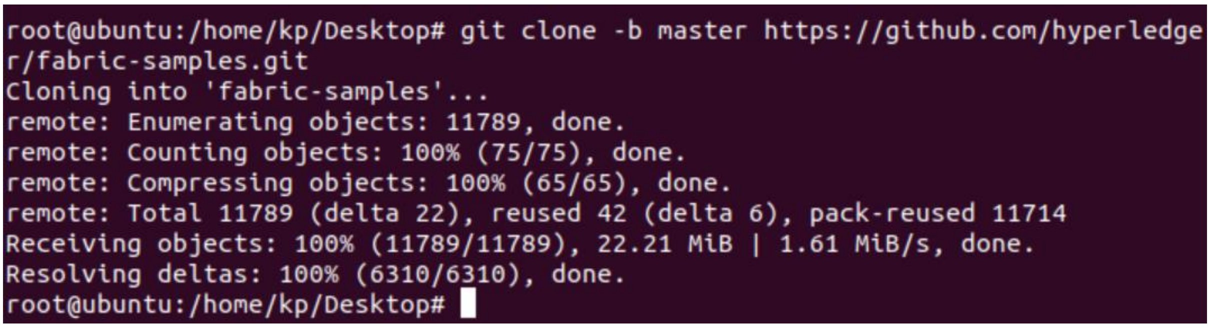
Working of Hyperledger Fabric

Necessary requirement for the project :



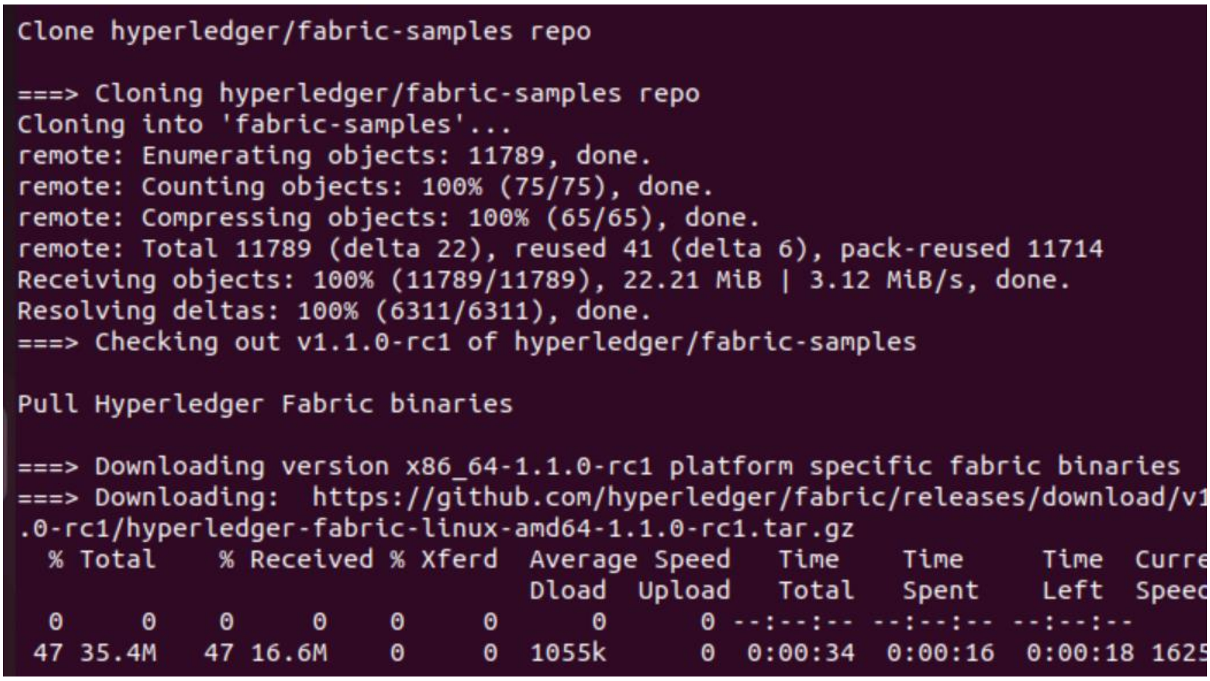
```
https://medium.com/cochain/hyperledger-fabric-on-windows-10-26723116c636

1)      sudo apt install aptitude
        sudo aptitude install npm



2)      git clone -b master https://github.com/hyperledger/fabric-samples.git
        cd fabric-samples
        git tag
        git checkout v1.1.0-rc1
        curl -sSL https://goo.gl/6wtTN5 | bash -s 1.1.0-rc1
```

- Cloning the fabric-samples.git into Desktop for fabric implementation



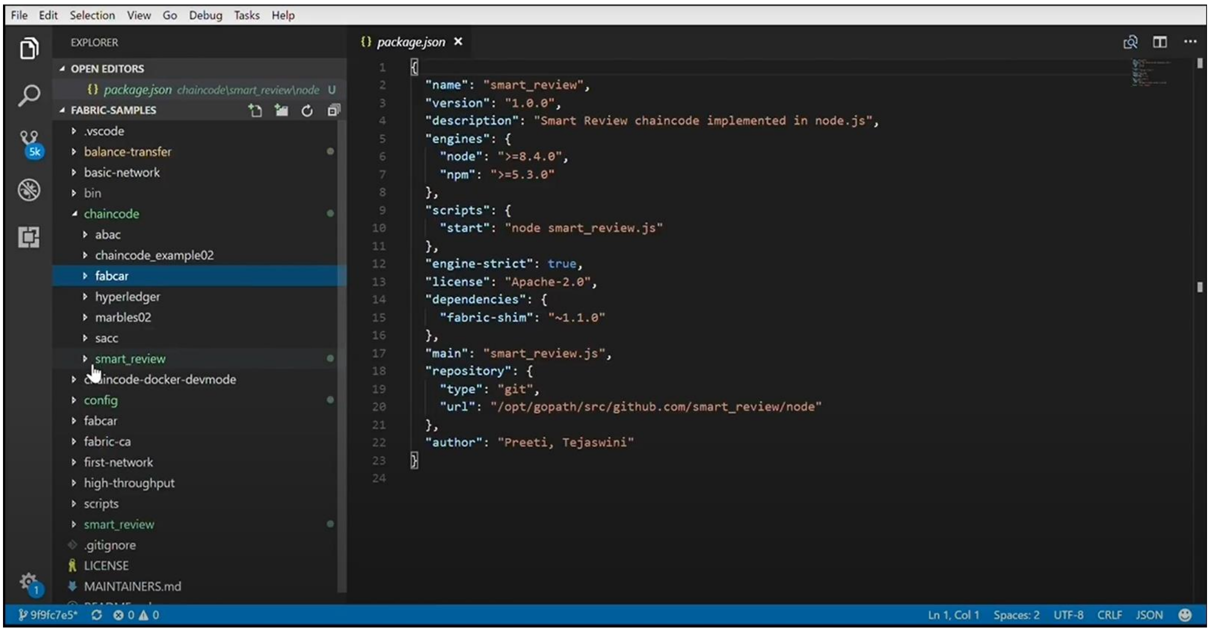
```
root@ubuntu:/home/kp/Desktop# git clone -b master https://github.com/hyperledge
r/fabric-samples.git
Cloning into 'fabric-samples'...
remote: Enumerating objects: 11789, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 11789 (delta 22), reused 42 (delta 6), pack-reused 11714
Receiving objects: 100% (11789/11789), 22.21 MiB | 1.61 MiB/s, done.
Resolving deltas: 100% (6310/6310), done.
root@ubuntu:/home/kp/Desktop# 
```

- Pulling fabric binaries



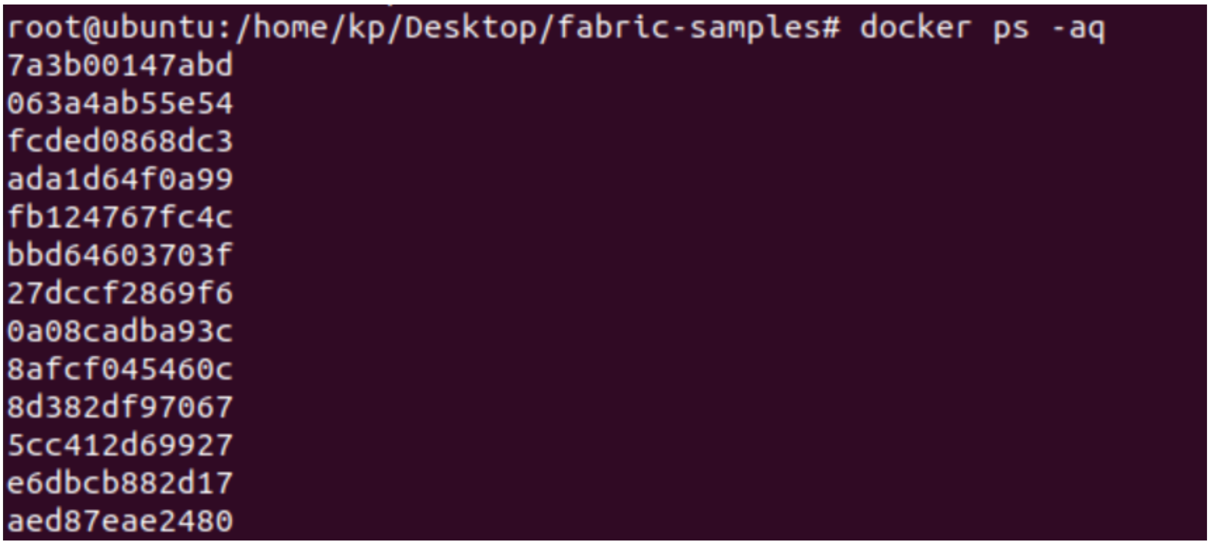
```
Clone hyperledger/fabric-samples repo

===> Cloning hyperledger/fabric-samples repo
Cloning into 'fabric-samples'...
remote: Enumerating objects: 11789, done.
remote: Counting objects: 100% (75/75), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 11789 (delta 22), reused 41 (delta 6), pack-reused 11714
Receiving objects: 100% (11789/11789), 22.21 MiB | 3.12 MiB/s, done.
Resolving deltas: 100% (6311/6311), done.
===> Checking out v1.1.0-rc1 of hyperledger/fabric-samples

Pull Hyperledger Fabric binaries

===> Downloading version x86_64-1.1.0-rc1 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v1
.0-rc1/hyperledger-fabric-linux-amd64-1.1.0-rc1.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Curre
                                 Dload  Upload   Total   Spent    Left  Speed
  0      0    0      0    0      0      0      0 --:--:-- --:--:-- --:--:--
 47 35.4M   47 16.6M    0      0  1055k      0  0:00:34  0:00:16  0:00:18 1625
```

- Fabric samples project in VS code



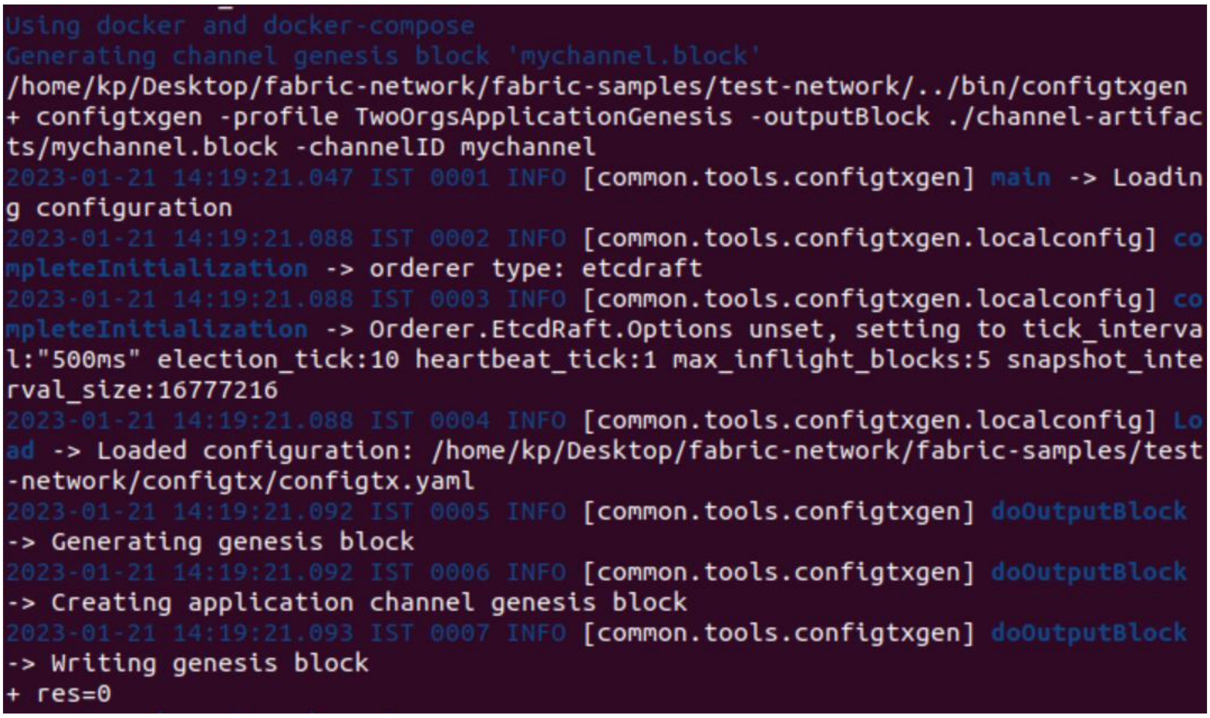
```
{
  "name": "smart_review",
  "version": "1.0.0",
  "description": "Smart Review chaincode implemented in node.js",
  "engines": {
    "node": ">=8.4.0",
    "npm": ">=5.3.0"
  },
  "scripts": {
    "start": "node smart_review.js"
  },
  "engine-strict": true,
  "license": "Apache-2.0",
  "dependencies": {
    "fabric-shim": "~1.1.0"
  },
  "main": "smart_review.js",
  "repository": {
    "type": "git",
    "url": "/opt/gopath/src/github.com/smart_review/node"
  },
  "author": "Preeti, Tejaswini"
}
```

- Docker Containers



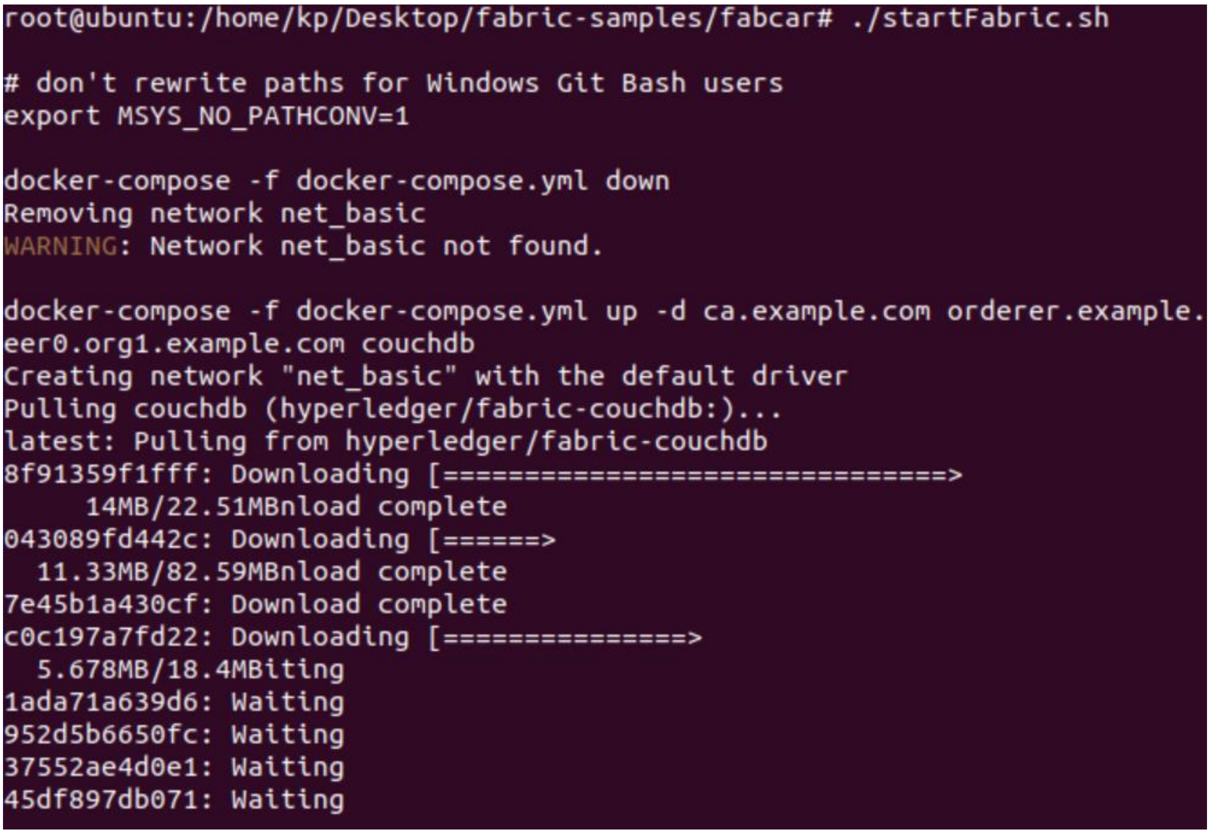
```
root@ubuntu:/home/kp/Desktop/fabric-samples# docker ps -aq
7a3b00147abd
063a4ab55e54
fcded0868dc3
ada1d64f0a99
fb124767fc4c
bbd64603703f
27dccf2869f6
0a08cadba93c
8afcf045460c
8d382df97067
5cc412d69927
e6dbcb882d17
aed87eae2480
```

- Genesis block is created successfully

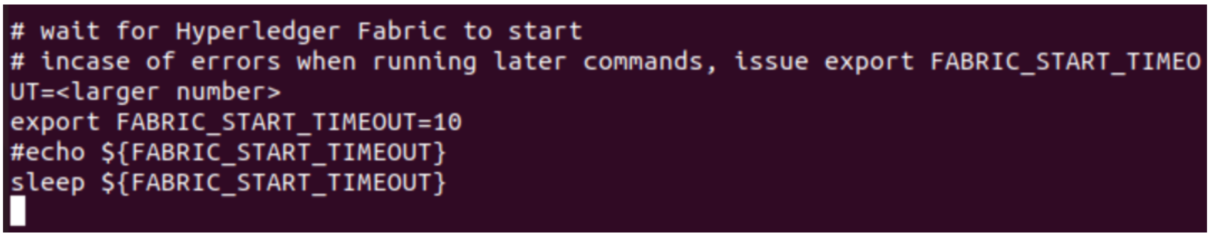


```
Using docker and docker-compose
Generating channel genesis block 'mychannel.block'
/home/kp/Desktop/fabric-network/fabric-samples/test-network/../bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifac
ts/mychannel.block -channelID mychannel
2023-01-21 14:19:21.047 IST 0001 INFO [common.tools.configtxgen] main -> Loadin
g configuration
2023-01-21 14:19:21.088 IST 0002 INFO [common.tools.configtxgen.localconfig] co
mpleteInitialization -> orderer type: etcdraft
2023-01-21 14:19:21.088 IST 0003 INFO [common.tools.configtxgen.localconfig] co
mpleteInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interva
l:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_inte
rval_size:16777216
2023-01-21 14:19:21.088 IST 0004 INFO [common.tools.configtxgen.localconfig] Lo
ad -> Loaded configuration: /home/kp/Desktop/fabric-network/fabric-samples/test
-network/configtx/configtx.yaml
2023-01-21 14:19:21.092 IST 0005 INFO [common.tools.configtxgen] doOutputBlock
-> Generating genesis block
2023-01-21 14:19:21.092 IST 0006 INFO [common.tools.configtxgen] doOutputBlock
-> Creating application channel genesis block
2023-01-21 14:19:21.093 IST 0007 INFO [common.tools.configtxgen] doOutputBlock
-> Writing genesis block
+ res=0
```

- The fabcar will start the car in go language in default which will also create a smart contract and add it to the network

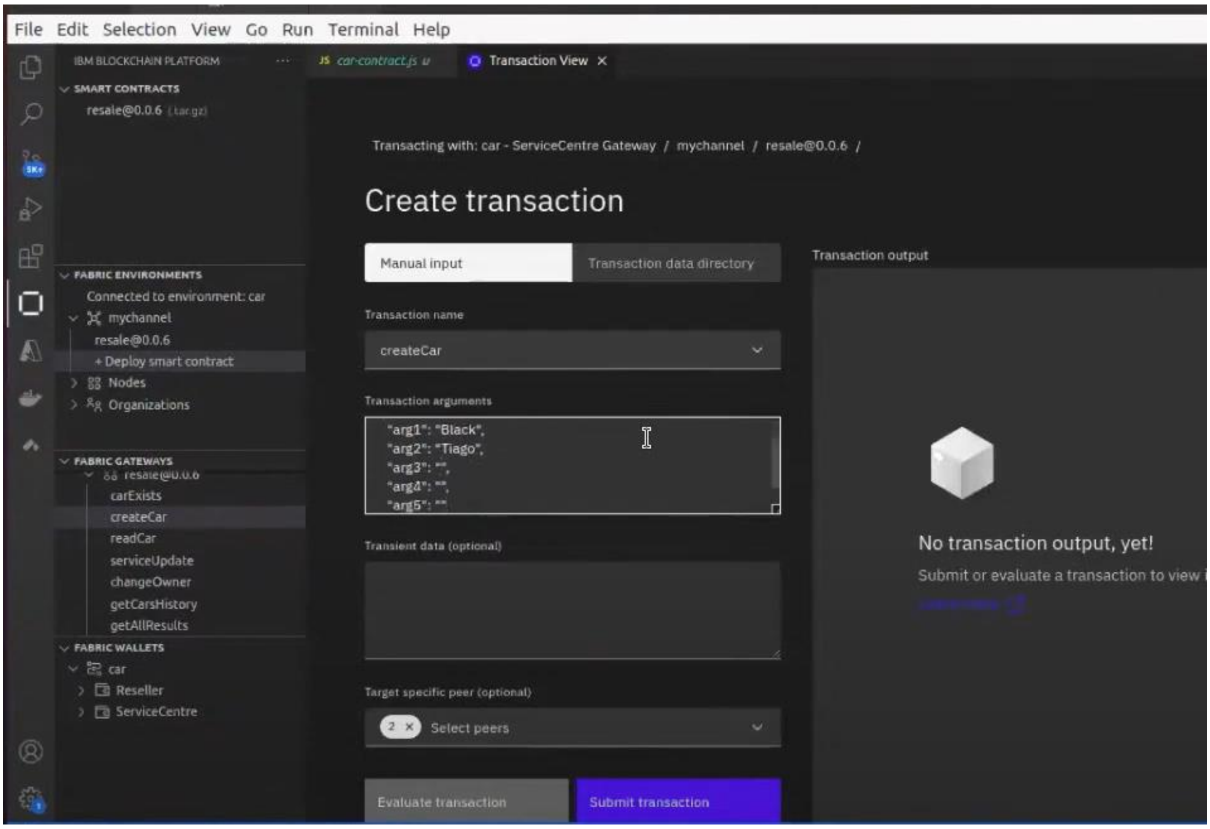


```
root@ubuntu:/home/kp/Desktop/fabric-samples/fabcar# ./startFabric.sh

# don't rewrite paths for Windows Git Bash users
export MSYS_NO_PATHCONV=1

docker-compose -f docker-compose.yml down
Removing network net_basic
WARNING: Network net_basic not found.

docker-compose -f docker-compose.yml up -d ca.example.com orderer.example.
eer0.org1.example.com couchdb
Creating network "net_basic" with the default driver
Pulling couchdb (hyperledger/fabric-couchdb:)...
latest: Pulling from hyperledger/fabric-couchdb
8f91359f1fff: Downloading [================================>
     14MB/22.51MBnload complete
043089fd442c: Downloading [======>
  11.33MB/82.59MBnload complete
7e45b1a430cf: Download complete
c0c197a7fd22: Downloading [===============>
  5.678MB/18.4MBiting
1ada71a639d6: Waiting
952d5b6650fc: Waiting
37552ae4d0e1: Waiting
45df897db071: Waiting
```



```
# wait for Hyperledger Fabric to start
# incase of errors when running later commands, issue export FABRIC_START_TIMEO
UT=<larger number>
export FABRIC_START_TIMEOUT=10
#echo ${FABRIC_START_TIMEOUT}
sleep ${FABRIC_START_TIMEOUT}
█
```

- Going on to the application



Transacting with: car - ServiceCentre Gateway / mychannel / resale@0.0.6 /

Create transaction

Transaction name: createCar

Transaction arguments:

```
"arg1": "Black",
"arg2": "Tiago",
"arg3": "",
"arg4": "",
"arg5": ""
```

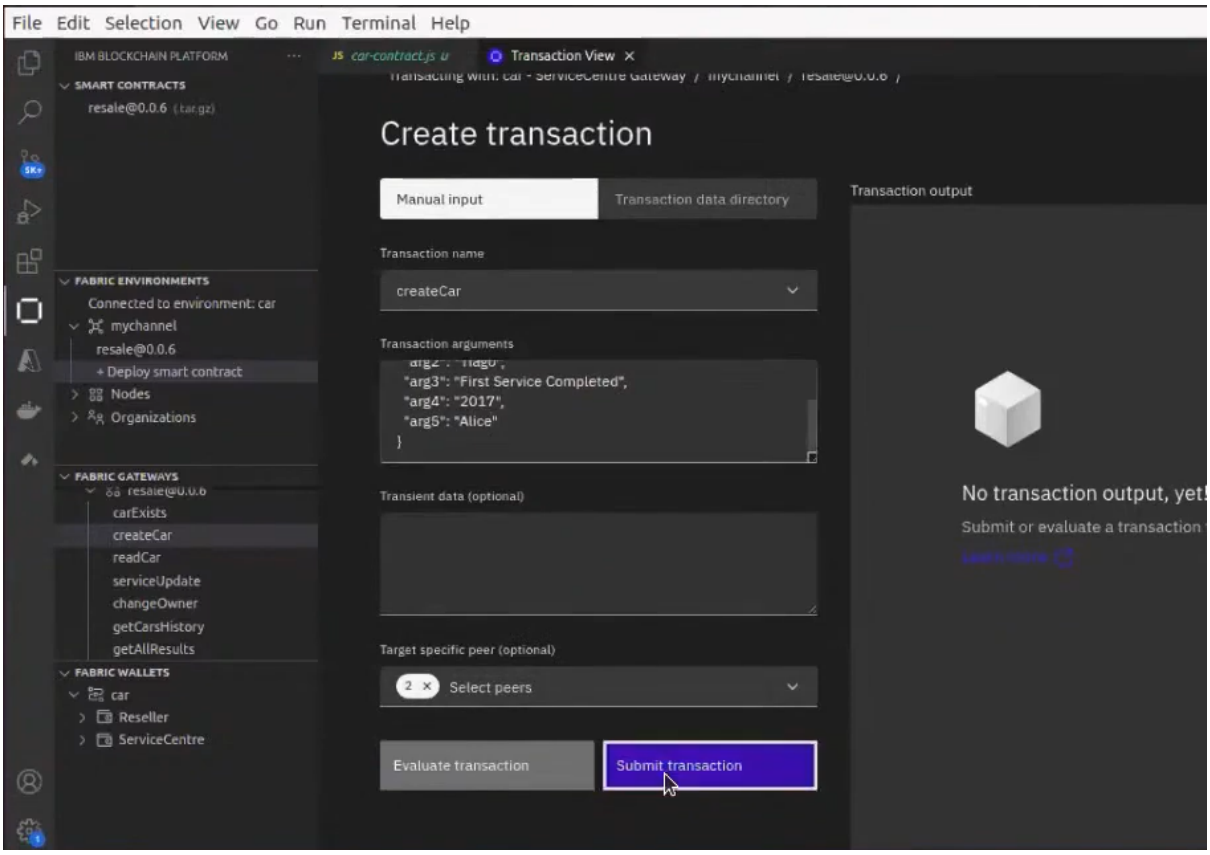
Transient data (optional)

Target specific peer (optional): 2 × Select peers

No transaction output, yet!

Submit or evaluate a transaction to view

- We are creating a new car using post method



Transacting with: car - ServiceCentre Gateway / mychannel / resale@0.0.6 /

Create transaction

Transaction name: createCar

Transaction arguments:

```
 "arg2": "Tiago",
 "arg3": "First Service Completed",
 "arg4": "2017",
 "arg5": "Alice"
}
```

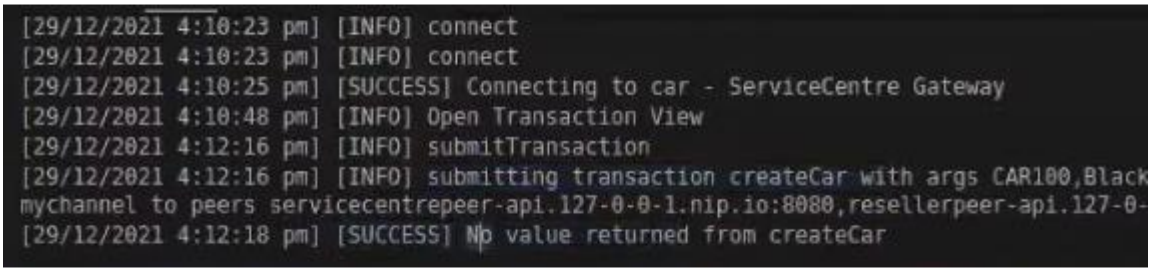
Transient data (optional)

Target specific peer (optional): 2 × Select peers

No transaction output, yet!

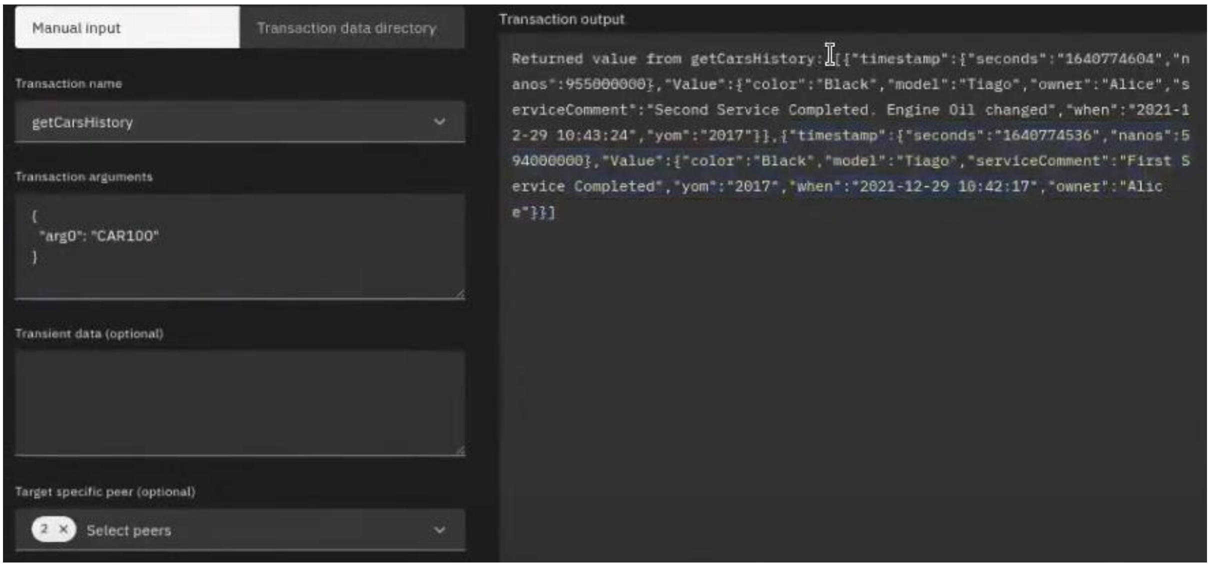
Submit or evaluate a transaction

- We got success message, the car created successfully



```
[29/12/2021 4:10:23 pm] [INFO] connect
[29/12/2021 4:10:23 pm] [INFO] connect
[29/12/2021 4:10:25 pm] [SUCCESS] Connecting to car - ServiceCentre Gateway
[29/12/2021 4:10:48 pm] [INFO] Open Transaction View
[29/12/2021 4:12:16 pm] [INFO] submitTransaction
[29/12/2021 4:12:16 pm] [INFO] submitting transaction createCar with args CAR100,Black
mychannel to peers servicecentrepeer-api.127-0-0-1.nip.io:8080,resellerpeer-api.127-0-
[29/12/2021 4:12:18 pm] [SUCCESS] No value returned from createCar
```

- At get request we will get the car



Transaction name: getCarsHistory

Transaction arguments:

```
{
  "arg0": "CAR100"
}
```

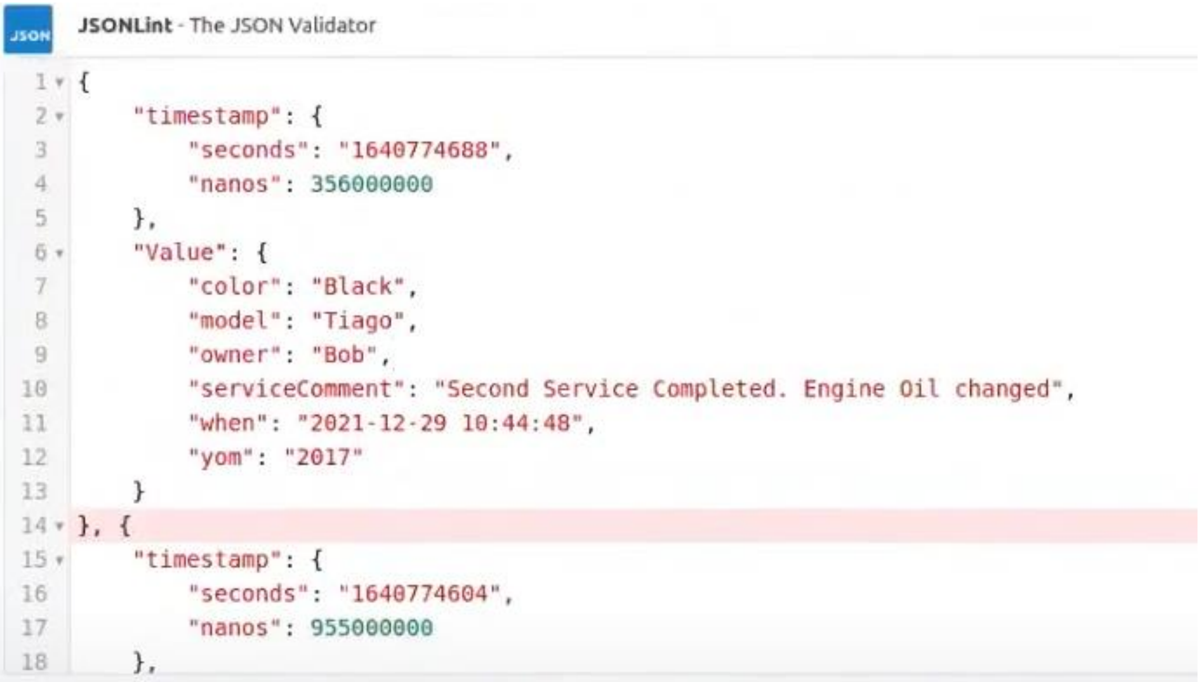
Transient data (optional):

Target specific peer (optional): 2 × Select peers

Transaction output

```
Returned value from getCarsHistory:[{"timestamp":{"seconds":"1640774604","n
anos":955000000},"Value":{"color":"Black","model":"Tiago","owner":"Alice","s
erviceComment":"Second Service Completed. Engine Oil changed","when":"2021-1
2-29 10:43:24","yom":"2017"}},{"timestamp":{"seconds":"1640774536","nanos":5
94000000},"Value":{"color":"Black","model":"Tiago","serviceComment":"First S
ervice Completed","yom":"2017","when":"2021-12-29 10:42:17","owner":"Alic
e"}}]
```

- Therefore blocks are created successfully, which is displayed in json format



```
{
    "timestamp": {
        "seconds": "1640774688",
        "nanos": 356000000
    },
    "Value": {
        "color": "Black",
        "model": "Tiago",
        "owner": "Bob",
        "serviceComment": "Second Service Completed. Engine Oil changed",
        "when": "2021-12-29 10:44:48",
        "yom": "2017"
    }
}, {
    "timestamp": {
        "seconds": "1640774604",
        "nanos": 955000000
    },
```

So we can create custom function and add on top of Hyperledger fabric and implement easily

Conclusion

These distinctive Hyperledger frameworks and tools demonstrate the enormous potential of Hyperledger for Blockchain technology.

These tools can be used to create industrial and non-monetary applications that are very scalable and reliable.

Blockchain technology is becoming more and more popular, and it has fundamentally altered how the technology sector looks for all time.

Also When taking into account Hyperledger Sawtooth's features and architecture, it can be said that the blockchain platform is simple for organisations in the financial, healthcare, retail, and other sectors to implement in order to create decentralised, secure, and reliable environments for their operations.