**TraXem Blockchain Manual**

|  |  |
| --- | --- |
| Công ty: | **VietSoftware International** |
| Người thực hiện: | **Lương Đức Duy** |
| Ngày gửi đề xuất: | **19/10/2020** |

MỤC LỤC

[1 Cài đặt TRAXEM BLOCKCHAIN 1](#_Toc54167238)

[1.1 Yêu cầu tiên quyết 1](#_Toc54167239)

[1.1.1 Môi trường cài đặt 1](#_Toc54167240)

[1.2 Cài đặt Hyperledger Binaries 1](#_Toc54167241)

[1.3 Cấu hình và triển khai mạng blockchain 1](#_Toc54167242)

[1.3.1 Crypto Material 1](#_Toc54167243)

[1.3.2 Mạng lưới Blockchain 2](#_Toc54167244)

[2 TRAXEM BLOCKCHAIN 3](#_Toc54167245)

[2.1 Start/Stop 3](#_Toc54167246)

[2.2 Backup and Restore 3](#_Toc54167247)

[2.3 Cấu hình Channel 3](#_Toc54167248)

[2.4 Chaincode 3](#_Toc54167249)

[3 TRAXEM BLOCKCHAIN APIs 5](#_Toc54167250)

[3.1 Triển khai 5](#_Toc54167251)

[3.2 API docs 5](#_Toc54167252)

[3.2.1 Get Production 5](#_Toc54167253)

[3.2.2 Create Production 5](#_Toc54167254)

[3.2.3 Update Production 5](#_Toc54167255)

[3.2.4 Get Block History Of Production 6](#_Toc54167256)

[3.2.5 Get Activity Log Of Production 6](#_Toc54167257)

[3.2.6 Get Activity Log 6](#_Toc54167258)

[3.2.7 Create Activity Log 6](#_Toc54167259)

# Cài đặt TRAXEM BLOCKCHAIN

## Yêu cầu tiên quyết

### Môi trường cài đặt

Traxem blockchain được thiết kế trên nền tảng Hyperledger Fabric v2.2 sẽ được triển khai trên hệ điều hành Ubuntu 18.04 và cần cài đặt thêm các module sau:

* Node v12.16.1
* NPM v6.13.4
* Docker v19.03.6
* Git repository traxem2-fabric-network

## Cài đặt Hyperledger Binaries

Đây là bộ hyperledger tool hỗ trợ generate các chi tiết cấu hình khởi tạo của hệ thống blockchain. Để cài bộ tool này sẽ cần chạy câu lệnh sau:

export PATH=<đường dẫn đến traxem2-fabric-network >/bin:$PATH

Lệnh trên sẽ add các tool của hyperledger vào system PATH để chúng ta có thể dễ dàng sử dụng chúng qua cửa sổ lệnh.

## Cấu hình và triển khai mạng blockchain

### Crypto Material

Đầu tiên chúng ta cần kiểm tra lại các file cấu hình được lưu ở trong thư mục config gồm 2 file configtx.yaml và crypto-config.yaml. Sau đó tại thư mục traxem2-fabric-network thực thi lệnh sau để thực hiện việc khởi tạo các chi tiết cho mạng lưới. Kết quả sẽ nằm ở thư mục crypto-config sau khi hoàn thành câu lệnh.

cryptogen generate --config=./config/crypto-config.yaml

Sau đó chúng ta cần thêm 2 biến môi trường mới cho hệ thống bằng cách thực thi:

- export FABRIC\_CFG\_PATH=$PWD/config

- export CHANNEL\_NAME=traxemchannel

Tiếp đến chúng ta sẽ tạo ra genesis block ( block đầu tiên của mạng lưới ), tạo cấu hình channel lần lượt theo thứ tự và nhận được kết quả ở thư mục channel-artifacts

- configtxgen -profile TraxemMultiNodeEtcdRaft --channelID byfn-sys-channel --outputBlock ./channel-artifacts/genesis.block

- configtxgen -profile OneOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID traxemchannel

- configtxgen -profile OneOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID traxemchannel -asOrg Org1MSP

Những chất liệu được tạo ra ở bước này sẽ được sử dụng để khởi chạy mạng lưới blockchain.

### Mạng lưới Blockchain

#### Cấu hình

Ở bước này ta cần tạo 1 file .env trong thư mục chính với nội dung như sau:

COMPOSE\_PROJECT\_NAME=traxem-network  
 IMAGE\_TAG=2.2  
 SYS\_CHANNEL=byfn-sys-channel  
 CA\_IMAGE\_TAG=1.4  
 COUCHDB\_IMAGE\_TAG=3.1.0

Kiểm tra các cấu hình cho docker trong thư mục base và các file: host1.yaml; host\_ca.yaml; host\_couch.yaml

#### Khởi chạy

Bắt đầu triển khai mạng lưới ta cần tạo 1 docker swarm và docker network để các container hoạt động với nhau:

docker swarm init  
 docker swarm join-token manager  
 docker network create --attachable --driver overlay traxem-network

Sau đó ta khởi chạy các container đã theo các file cấu hình của mạng lưới:

docker-compose -f host1.yaml -f host\_couch.yaml -f host\_ca.yaml up -d

Như vậy là mạng lưới đã bước đầu hoạt động. Ở phần tiếp theo chúng ta sẽ thực hiện việc các thao tác chính với mạng lưới

# TRAXEM BLOCKCHAIN

## Start/Stop

Để khởi động hoặc dừng mạng blockchain thì trong thư mục scripts đã có 2 sh script là startNetwork.sh và stopNetwork.sh giúp chúng ta thao tác nhanh việc bật và tắt. Về bản chất thì 2 scripts trên sử dụng docker-compose để start và stop các file docker yaml ở bước trước.

## Backup and Restore

#### Backup

Sao lưu và phục hồi một mạng Traxem blockchain sẽ cần chung ta sao lưu 4 dữ liệu sau:

- Crypto materials (Public/Private Keys và Certificates) được generate ra trong thư mục crypto-config

- Channel artifacts (Genesis block và channel block) có trong thư mục channel-artifacts

- Backup files của peer (/var/hyperledger/production)

- Backup files của Orderer (/var/hyperledger/production/orderer/)

Với những dữ liệu trên ta có thể khởi tạo mạng lưới kể cả khi nó hoàn toàn sụp đổ.

#### Restore

Để phục hồi ta sẽ cần ghi đè các dữ liệu đã được backup vào các thư mục tương ứng của crypto-config, channel-artifacts và peer, orderer docker containers. Sau đó khởi chạy lại network như bước trước.

## Cấu hình Channel

Sau khi khởi chạy mạng lưới blockchain, chúng ta cần tạo channel chung cho các peer của các tổ chức cùng hoạt động.

docker exec -it cli peer channel create -o orderer.traxem.vn:7050 -c traxemchannel -f ./channel-artifacts/channel.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/traxem.vn/orderers/orderer.traxem.vn/msp/tlscacerts/tlsca.traxem.vn-cert.pem

Tiếp theo, lần lượt thêm các peer vào channel vừa tạo

docker exec -it cli peer channel join -b traxemchannel.block

docker exec -it -e CORE\_PEER\_ADDRESS="peer1.org1.traxem.vn:8051" -e CORE\_PEER\_TLS\_ROOTCERT\_FILE="/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.traxem.vn/peers/peer1.org1.traxem.vn/tls/ca.crt" cli peer channel join -b traxemchannel.block

## Chaincode

Chaincode có quy trình cài đặt lần lượt từ việc đóng gói, cài đặt lên mạng lưới, các org vào approve và khi đạt đủ sự đồng thuận thì 1 org bất kì có thể commit để chaincode đi vào hoạt động. Các quy trình này đã được gói lại bằng 2 scripts trong thư mục deploy-chaincode. Thực thi lần lượt:

bash ./deploy-chaincode/install.sh <cc\_folder> <cc\_label>

bash ./deploy-chaincode/approveAndCommit.sh <chaincode\_name> <package> <version> <sequence>

Khi cần nâng cấp chaincode, ta sẽ cập nhật source code trong thư mục chaincode và nâng version của nó trong file package.json đồng thời nâng sequence khi chạy lệnh approve and commit.

# TRAXEM BLOCKCHAIN APIs

## Triển khai

Tại thư mục server/api ta tiến hành cài đặt npm packages:

npm install

Sau đó khởi chạy api server với lệnh: (kết hợp sử dụng pm2 hoặc screen)

npm start

## API docs

### Get Production

Method: GET

Path: <host>/production/:id

Param: id - id của production

### Create Production

Method: POST

Path: <host>/production/

Param: none

Body: id: string*()*.required*()*,  
 barcode: string*()*.required*()*,  
 createdTime: date*()*.required*()*,  
 createdId: string*()*.required*()*,  
 description: string*()*.notRequired*()*,  
 expiryDate: date*()*.notRequired*()*,  
 name: string*()*.required*()*,  
 producedDate: date*()*.notRequired*()*,  
 qrcode: string*()*.required*()*,  
 qrcode2: string*()*.required*()*,  
 quantityUnit: string*()*.notRequired*()*,  
 salcode: string*()*.required*()*,  
 sapcode: string*()*.required*()*,  
 status: string*()*.notRequired*()*,  
 updatedTime: date*()*.notRequired*()*,  
 updatedId: string*()*.notRequired*()*,  
 warrantyInMonths: number*()*.notRequired*()*,  
 warrantyRemarks: string*()*.notRequired*()*,  
 activatingPoint: number*()*.notRequired*()*,  
 viewTimes: number*()*.notRequired*()*,

### Update Production

Method: PUT

Path: <host>/production/

Param: none

Body: id: string*()*.required*()*,  
 barcode: string*()*.required*()*,  
 createdTime: date*()*.required*()*,  
 createdId: string*()*.required*()*,  
 description: string*()*.notRequired*()*,  
 expiryDate: date*()*.notRequired*()*,  
 name: string*()*.required*()*,  
 producedDate: date*()*.notRequired*()*,  
 qrcode: string*()*.required*()*,  
 qrcode2: string*()*.required*()*,  
 quantityUnit: string*()*.notRequired*()*,  
 salcode: string*()*.required*()*,  
 sapcode: string*()*.required*()*,  
 status: string*()*.notRequired*()*,  
 updatedTime: date*()*.notRequired*()*,  
 updatedId: string*()*.notRequired*()*,  
 warrantyInMonths: number*()*.notRequired*()*,  
 warrantyRemarks: string*()*.notRequired*()*,  
 activatingPoint: number*()*.notRequired*()*,  
 viewTimes: number*()*.notRequired*()*,

### Get Block History Of Production

Method: GET

Path: <host>/production/:id/block\_history

Param: id - id của production

### Get Activity Log Of Production

Method: GET

Path: <host>/production/:id/log

Param: id - id của production

### Get Activity Log

Method: GET

Path: <host>/activity\_log/:id

Param: id - id của activity log

### Create Activity Log

Method: POST

Path: <host>/activity\_log/

Param: none

Body: id: string*()*.required*()*,  
 content: string*()*.required*()*,  
 createdTime: date*()*.required*()*,  
 createdId: string*()*.required*()*,  
 productionId: array*()*.notRequired*()*,  
 productionName: string*()*.notRequired*()*,