**Integration of Hyperledger explorer with fabric and integration of Hyperledger composer with fabric**

Hyperledger Explorer is a blockchain module and one of the Hyperledger projects hosted by The Linux Foundation. Designed to create a user-friendly Web application, Hyperledger Explorer can view, invoke, deploy or query blocks, transactions and associated data, network information (name, status, list of nodes), chain codes and transaction families, as well as any other relevant information stored in the ledger. Hyperledger Explorer was initially contributed by IBM, Intel and DTCC. Check the current status of Hyperledger Explorer.

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. Our primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems. You can use Composer to rapidly develop use cases and deploy a blockchain solution in weeks rather than months. Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.

**Steps to Implement the above :-**

**cd`**~

**sudo** apt update **&&** **sudo** apt upgrade

**sudo** apt **install** **git** **make** **gcc** **g++** libltdl-dev curl python pkg-config

**ifconfig**

**sudo** reboot

Install Docker**/** Docker Compose

curl -fsSL test.docker.com **|** **sh**

**sudo** usermod -aG docker $USER

**exec** **sudo** **su** -l $USER

**sudo** curl -L https:**//**github.com**/**docker**/**compose**/**releases**/**download**/**1.21.2**/**docker-compose-$**(uname** -s**)**-$**(uname** -m**)** -o **/**usr**/**local**/**bin**/**docker-compose

**sudo** **chmod** +x **/**usr**/**local**/**bin**/**docker-compose

Install nodejs **/** Composer

**wget** https:**//**nodejs.org**/**dist**/**v8.11.2**/**node-v8.11.2-linux-x64.tar.xz

**tar** -xf node-v8.11.2-linux-x64.tar.xz

**mv** node-v8.11.2-linux-x64**/** node**/**

**echo** 'export PATH=~/node/bin:$PATH' **>>** ~**/**.profile

**source** .profile

npm **install** -g npm

npm **install** -g grpc

npm **install** -g composer-cli composer-playground generator-hyperledger-composer composer-rest-server yo

Install Go **/** Hyperledger Binaries

**wget** https:**//**dl.google.com**/**go**/**go1.10.2.linux-amd64.tar.gz

**sudo** **tar** -C **/**usr**/local** -xvf go1.10.2.linux-amd64.tar.gz

**echo** 'export GOPATH=/opt/go' **>>** ~**/**.profile

**echo** 'export GOBIN=/opt/go/bin' **>>** ~**/**.profile

**echo** 'export PATH=/usr/local/go/bin:$PATH' **>>** ~**/**.profile

**source** ~**/**.profile

**sudo** **mkdir** -p **/**opt**/**go**/**bin

**sudo** **mkdir** **/**opt**/**go**/**src

**cd** $GOPATH

**sudo** **chown** -R $USER **/**opt**/**go

**cd** src

**mkdir** -p github.com**/**hyperledger

**cd** github.com**/**hyperledger

**git clone** https:**//**gerrit.hyperledger.org**/**r**/**fabric

**cd** fabric

**sudo** apt **install** golang-ginkgo-dev

**make** dist-clean all **(**repeat this **until** you get no errors**)**

All steps of https:**//**hyperledger.github.io**/**composer**/**latest**/**tutorials**/**deploy-to-fabric-multi-org.html

**cd** ~

**git clone** -b issue-6978 https:**//**github.com**/**sstone1**/**fabric-samples.git

**cd** fabric-samples

curl -sSL https:**//**goo.gl**/**6wtTN5 **|** **bash** -s 1.1.0

**echo** 'export PATH=~/fabric-samples/bin:$PATH' **>>** ~**/**.profile

**source** ~**/**.profile

**cd** first-network

.**/**byfn.sh -m generate

.**/**byfn.sh -m up -s couchdb -a

open TERMINAL 2 **for** next steps:

**mkdir** -p **/**tmp**/**composer**/**org1

**mkdir** -p **/**tmp**/**composer**/**org2

**cd** **/**tmp**/**composer

**nano** byfn-network.json

**{**

    "name": "byfn-network",

    "x-type": "hlfv1",

    "version": "1.0.0",

    "channels": **{**

        "mychannel": **{**

            "orderers": **[**

                "orderer.example.com"

**]**,

            "peers": **{**

                "peer0.org1.example.com": **{**

                    "endorsingPeer": **true**,

                    "chaincodeQuery": **true**,

                    "eventSource": **true**

**}**,

                "peer1.org1.example.com": **{**

                    "endorsingPeer": **true**,

                    "chaincodeQuery": **true**,

                    "eventSource": **true**

**}**,

                "peer0.org2.example.com": **{**

                    "endorsingPeer": **true**,

                    "chaincodeQuery": **true**,

                    "eventSource": **true**

**}**,

                "peer1.org2.example.com": **{**

                    "endorsingPeer": **true**,

                    "chaincodeQuery": **true**,

                    "eventSource": **true**

**}**

**}**

**}**

**}**,

    "organizations": **{**

        "Org1": **{**

            "mspid": "Org1MSP",

            "peers": **[**

                "peer0.org1.example.com",

                "peer1.org1.example.com"

**]**,

            "certificateAuthorities": **[**

                "ca.org1.example.com"

**]**

**}**,

        "Org2": **{**

            "mspid": "Org2MSP",

            "peers": **[**

                "peer0.org2.example.com",

                "peer1.org2.example.com"

**]**,

            "certificateAuthorities": **[**

                "ca.org2.example.com"

**]**

**}**

**}**,

    "orderers": **{**

        "orderer.example.com": **{**

            "url": "grpcs://localhost:7050",

            "grpcOptions": **{**

                "ssl-target-name-override": "orderer.example.com"

**}**,

            "tlsCACerts": **{**

                "pem": "INSERT\_ORDERER\_CA\_CERT"

**}**

**}**

**}**,

    "peers": **{**

        "peer0.org1.example.com": **{**

            "url": "grpcs://localhost:7051",

            "eventUrl": "grpcs://localhost:7053",

            "grpcOptions": **{**

                "ssl-target-name-override": "peer0.org1.example.com"

**}**,

            "tlsCACerts": **{**

                "pem": "INSERT\_ORG1\_CA\_CERT"

**}**

**}**,

        "peer1.org1.example.com": **{**

            "url": "grpcs://localhost:8051",

            "eventUrl": "grpcs://localhost:8053",

            "grpcOptions": **{**

                "ssl-target-name-override": "peer1.org1.example.com"

**}**,

            "tlsCACerts": **{**

                "pem": "INSERT\_ORG1\_CA\_CERT"

**}**

**}**,

        "peer0.org2.example.com": **{**

            "url": "grpcs://localhost:9051",

            "eventUrl": "grpcs://localhost:9053",

            "grpcOptions": **{**

                "ssl-target-name-override": "peer0.org2.example.com"

**}**,

            "tlsCACerts": **{**

                "pem": "INSERT\_ORG2\_CA\_CERT"

**}**

**}**,

        "peer1.org2.example.com": **{**

            "url": "grpcs://localhost:10051",

            "eventUrl": "grpcs://localhost:10053",

            "grpcOptions": **{**

                "ssl-target-name-override": "peer1.org2.example.com"

**}**,

            "tlsCACerts": **{**

                "pem": "INSERT\_ORG2\_CA\_CERT"

**}**

**}**

**}**,

    "certificateAuthorities": **{**

        "ca.org1.example.com": **{**

            "url": "https://localhost:7054",

            "caName": "ca-org1",

            "httpOptions": **{**

                "verify": **false**

**}**

**}**,

        "ca.org2.example.com": **{**

            "url": "https://localhost:8054",

            "caName": "ca-org2",

            "httpOptions": **{**

                "verify": **false**

**}**

**}**

**}**

**}**

open TERMINAL 3 **for** next steps:

**cd** ~**/**fabric-samples**/**first-network

**awk** 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' crypto-config**/**ordererOrganizations**/**example.com**/**orderers**/**orderer.example.com**/**tls**/**ca.crt **>/**tmp**/**composer**/**ca-orderer.txt

**cat** **/**tmp**/**composer**/**ca-orderer.txt

**//**insert into byfn-network.json where it says “INSERT\_ORDERER\_CA\_CERT”

**awk** 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' crypto-config**/**peerOrganizations**/**org1.example.com**/**peers**/**peer0.org1.example.com**/**tls**/**ca.crt **>/**tmp**/**composer**/**org1**/**ca-org1.txt

**cat** **/**tmp**/**composer**/**org1**/**ca-org1.txt

**//**insert into byfn-network.json where it says “INSERT\_ORG1\_CA\_CERT”

**awk** 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' crypto-config**/**peerOrganizations**/**org2.example.com**/**peers**/**peer0.org2.example.com**/**tls**/**ca.crt **>/**tmp**/**composer**/**org2**/**ca-org2.txt

**cat** **/**tmp**/**composer**/**org2**/**ca-org2.txt

**//**insert into byfn-network.json where it says “INSERT\_ORG2\_CA\_CERT”

save byfn-network.json and quit **in** TERMINAL 2, CONTINUE IN TERMINAL 2

**cp** **/**tmp**/**composer**/**byfn-network.json **/**tmp**/**composer**/**org1**/**byfn-network-org1.json

**cp** **/**tmp**/**composer**/**byfn-network.json **/**tmp**/**composer**/**org2**/**byfn-network-org2.json

**cd** org1

**nano** byfn-network-org1.json

**//**after the line with “version”, add the following lines of code:

 "client": **{**

        "organization": "Org1",

        "connection": **{**

            "timeout": **{**

                "peer": **{**

                    "endorser": "300",

                    "eventHub": "300",

                    "eventReg": "300"

**}**,

                "orderer": "300"

**}**

**}**

**}**,

**cd** ..**/**org2

**nano** byfn-network-org2.json

**//**after the line with “version”, add the following lines of code:

 "client": **{**

        "organization": "Org2",

        "connection": **{**

            "timeout": **{**

                "peer": **{**

                    "endorser": "300",

                    "eventHub": "300",

                    "eventReg": "300"

**}**,

                "orderer": "300"

**}**

**}**

**}**,

switch to TERMINAL 3 **for** next steps:

**export** ORG1=crypto-config**/**peerOrganizations**/**org1.example.com**/**users**/**Admin**@**org1.example.com**/**msp

**cp** -p $ORG1**/**signcerts**/**A**\***.pem **/**tmp**/**composer**/**org1

**cp** -p $ORG1**/**keystore**/\***\_sk **/**tmp**/**composer**/**org1

**export** ORG2=crypto-config**/**peerOrganizations**/**org2.example.com**/**users**/**Admin**@**org2.example.com**/**msp

**cp** -p $ORG2**/**signcerts**/**A**\***.pem **/**tmp**/**composer**/**org2

**cp** -p $ORG2**/**keystore**/\***\_sk **/**tmp**/**composer**/**org2

composer card create -p **/**tmp**/**composer**/**org1**/**byfn-network-org1.json -u PeerAdmin -c **/**tmp**/**composer**/**org1**/**Admin**@**org1.example.com-cert.pem -k**/**tmp**/**composer**/**org1**/\***\_sk -r PeerAdmin -r ChannelAdmin -f PeerAdmin**@**byfn-network-org1.card

composer card create -p **/**tmp**/**composer**/**org2**/**byfn-network-org2.json -u PeerAdmin -c **/**tmp**/**composer**/**org2**/**Admin**@**org2.example.com-cert.pem -k**/**tmp**/**composer**/**org2**/\***\_sk -r PeerAdmin -r ChannelAdmin -f PeerAdmin**@**byfn-network-org2.card

composer card import -f PeerAdmin**@**byfn-network-org1.card --card PeerAdmin**@**byfn-network-org1

composer card import -f PeerAdmin**@**byfn-network-org2.card --card PeerAdmin**@**byfn-network-org2

**git clone** https:**//**github.com**/**hyperledger**/**composer-sample-networks

**cd** composer-sample-networks

npm **install** -g lerna

lerna bootstrap

**cd** ..

**cp** composer-sample-networks**/**packages**/**trade-network**/**dist**/**trade-network.bna .

composer network **install** --card PeerAdmin**@**byfn-network-org1 --archiveFile trade-network.bna

composer network **install** --card PeerAdmin**@**byfn-network-org2 --archiveFile trade-network.bna

switch to TERMINAL 2 **for** next steps:

**cd** **/**tmp**/**composer

**nano** endorsement-policy.json **//**add the following lines

**{**

    "identities": **[**

**{**

            "role": **{**

                "name": "member",

                "mspId": "Org1MSP"

**}**

**}**,

**{**

            "role": **{**

                "name": "member",

                "mspId": "Org2MSP"

**}**

**}**

**]**,

    "policy": **{**

        "2-of": **[**

**{**

                "signed-by": 0

**}**,

**{**

                "signed-by": 1

**}**

**]**

**}**

**}**

switch to TERMINAL 3 **for** next steps:

composer identity request -c PeerAdmin**@**byfn-network-org1 -u admin -s adminpw -d alice

composer identity request -c PeerAdmin**@**byfn-network-org2 -u admin -s adminpw -d bob

composer network start -c PeerAdmin**@**byfn-network-org1 -n trade-network -V 0.2.6 -o endorsementPolicyFile=**/**tmp**/**composer**/**endorsement-policy.json -A alice -C alice**/**admin-pub.pem -A bob -C bob**/**admin-pub.pem

composer card create -p **/**tmp**/**composer**/**org1**/**byfn-network-org1.json -u alice -n trade-network -c alice**/**admin-pub.pem -k alice**/**admin-priv.pem

composer card import -f alice**@**trade-network.card

composer network **ping** -c alice**@**trade-network

composer card create -p **/**tmp**/**composer**/**org2**/**byfn-network-org2.json -u bob -n trade-network -c bob**/**admin-pub.pem -k bob**/**admin-priv.pem

composer card import -f bob**@**trade-network.card

composer network **ping** -c bob**@**trade-network

composer-playground -p 8181 or **nohup** composer-playground -p 8181 **&** to background the task

switch to TERMINAL 2 **for** next steps:

**cd** ~

**sudo** apt **install** postgresql postgresql-contrib

**git clone** https:**//**github.com**/**hyperledger**/**blockchain-explorer.git

**cd** blockchain-explorer

**sudo** -u postgres psql

\i app**/**db**/**explorerpg.sql

\i app**/**db**/**updatepg.sql

\q

**nano** config.json **(**change fabric-path to where your fabric samples are contained.**)**

npm **install**

**cd** app**/test**

npm **install**

npm run **test**

**cd** ..**/**..**/**client

npm **install**

npm **test** -- -u –coverage

npm run build

**cd** ..

.**/**start.sh