**Adding new Organization dynamically**

Reference: <https://hyperledger-fabric.readthedocs.io/en/release-1.4/channel_update_tutorial.html>

**Make sure that existing network is up and running 3 Orgs and working properly (End to end: starting from Artifacts generation to Chaincodes invocation).**

Create new VM and clone <https://github.com/konda-kalyan/HLF-Multi-Host-Docker-Swarm>

cd to ‘~/HLF-Multi-Host-Docker-Swarm/network’ and Switch to ‘Add-new-Org’

**Execute all steps on Manager node only until unless specify do it on New node (let’s call it as Org4 node)**

### Generate crypto info for Org4 (with 2 peers for it)

* + - * 1. Create yaml files

Create ‘*org4-artifacts*’ dir and *cd* to that dir.

Create ‘org4-crypto.yaml’ and ‘configtx.yaml’ with respective configurations. Take reference from ‘<https://github.com/hyperledger/fabric-samples/tree/release-1.4/first-network/org3-artifacts>’ files

* + - * 1. Run cryptogen tool to generate crypto artifacts

*rm -rf crypto-config; cryptogen generate --config=./org4-crypto.yaml*

* + - * 1. Now use the configtxgen utility to print out the Org4-specific configuration material in JSON

*export FABRIC\_CFG\_PATH=$PWD && ../bin/configtxgen -printOrg Org4MSP > ../channel-artifacts/org4.json*

* + - * 1. Copy existing Orderer crypto artifacts to Org4’s crypto-config dir

*cd ../ && cp -r crypto-config/ordererOrganizations org4-artifacts/crypto-config/*

### Prepare Channel configurations for Org4

* + - * 1. Prepare the **Org1** CLI Environment

Login into **Org1** CLI container

*docker exec -it* *hlf\_services\_cli.1.7xtgqo5ezqo5x1hu31lhpzi3t bash*

**ALL BELOW COMMANDS HAVE TO BE RUN WITH IN CLI CONTAINTER**

Export the ORDERER\_CA and CHANNEL\_NAME variables

*export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer1.example.com/msp/tlscacerts/tlsca.example.com-cert.pem && export CHANNEL\_NAME=mychannel*

*echo $ORDERER\_CA && echo $CHANNEL\_NAME*

* + - * 1. Fetch the **latest** configuration block

*peer channel fetch config config\_block.pb -o orderer1.example.com:7050 -c $CHANNEL\_NAME --tls --cafile $ORDERER\_CA*

* + - * 1. Convert the config block to JSON and trim it down using **jq** tool

*configtxlator proto\_decode --input config\_block.pb --type common.Block | jq .data.data[0].payload.data.config > config.json*

* + - * 1. Add the Org4 Crypto material to the above config block and prepare channel config data

**From outside of CLI container (better do it from other terminal)**, copy ‘./channel-artifacts/org4.json’ to CLI container working dir i.e /opt/gopath/src/github.com/hyperledger/fabric/peer

*docker cp ./channel-artifacts/org4.json hlf\_services\_cli.1.7xtgqo5ezqo5x1hu31lhpzi3t:/opt/gopath/src/github.com/hyperledger/fabric/peer*

**Now, again start executing below commands from CLI container**

Get what has been modified

*jq -s '.[0] \* {"channel\_group":{"groups":{"Application":{"groups": {"Org4MSP":.[1]}}}}}' config.json org4.json > modified\_config.json*

Translate config.json back into a protobuf called config.pb

*configtxlator proto\_encode --input config.json --type common.Config --output config.pb*

Encode modified\_config.json to modified\_config.pb

*configtxlator proto\_encode --input modified\_config.json --type common.Config --output modified\_config.pb*

Calculate the delta between these two config protobufs

*configtxlator compute\_update --channel\_id $CHANNEL\_NAME --original config.pb --updated modified\_config.pb --output org4\_update.pb*

Decode *org4\_update.pb* into editable JSON format and call it org4\_update.json

*configtxlator proto\_decode --input org4\_update.pb --type common.ConfigUpdate | jq . > org4\_update.json*

Wrap above json into an envelope message

*echo '{"payload":{"header":{"channel\_header":{"channel\_id":"'$CHANNEL\_NAME'", "type":2}},"data":{"config\_update":'$(cat org4\_update.json)'}}}' | jq . > org4\_update\_in\_envelope.json*

Covert back to pb form

*configtxlator proto\_encode --input org4\_update\_in\_envelope.json --type common.Envelope --output org4\_update\_in\_envelope.pb*

### Get signatures from other 3 peers and submit channel update transaction

Continue from same Org1 CLI container session only

* + - * 1. Get sign from Org1 Admin (Remember that the CLI container is bootstrapped with the Org1 MSP material, so we simply need to issue the).

*peer channel signconfigtx -f org4\_update\_in\_envelope.pb*

* + - * 1. Get sign from Org2 Admin

First, set environment variables of Org2

*export CORE\_PEER\_LOCALMSPID="Org2MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051*

Now, get sign

*peer channel signconfigtx -f org4\_update\_in\_envelope.pb*

* + - * 1. Issue Channel update command from Org3 point of view (by Fabric functionality, Org3 automatically adds its signature before issuing update command. So, no need to explicitly run sign command)

First, set environment variables of Org3

*export CORE\_PEER\_LOCALMSPID="Org3MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051*

Now, issue channel update command

*peer channel* ***update*** *-f org4\_update\_in\_envelope.pb -c $CHANNEL\_NAME -o orderer1.example.com:7050 --tls --cafile $ORDERER\_CA*

### Prepare CC package with new version

Continue from same Org1 CLI container session only

*peer chaincode package -l java -n couchdb\_java\_simple -v v****1*** *-p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries couchdb\_java\_simple\_v1****.pak***

### Bring up Org4 related containers

**Run below commands from normal Manager node Linux terminal (not from CLI container). Run from Manager node only.**

1. Copy CC package file from Org1 CLI container to Manager node terminal

*docker cp hlf\_services\_cli.1.791bvwfnizp11vp6jy1v8jvd0:/opt/gopath/src/github.com/hyperledger/fabric/peer/couchdb\_java\_simple\_v3.pak ~/HLF-Multi-Host-Docker-Swarm/network*

1. Copy artifacts to Org4 node

Change ip address of Org4 in below script and run it

*./copy\_org4\_crypto\_and\_scp\_other\_nodes.sh*

1. Copy CC package file to Org4 node

*scp couchdb\_java\_simple\_v1.pak ubuntu@ip-172-31-37-174:~/HLF-Multi-Host-Docker-Swarm/network*

1. Prepare docker-compose files for Peers, Orderer, Couchdb, CA and CLI and place them in ‘org4’ dir

*mkdir org4; cd org4*

*vim docker-compose-peer.yml*

*vim docker-compose-orderer.yml*

*vim docker-compose-services.yml* (In this file, don’t forget to update CA’s ‘\*\_sk’ file reference for org4 ca service)

1. Run all Org4 services (bring up containers)
   * + - 1. Get docker swarm token to join Org4 to network

*docker swarm join-token worker*

* + - * 1. Join Org4 to docker swarm network **(Run this command alone from Org4 node)**

*docker swarm leave -f;*

*docker swarm join --token SWMTKN-1-29dq4k297zd55u3akzhttvxkdqb37urdebco0jgfels1rmcw9g-6qu174yeb8vlv9hwgd3k4z7sv 172.31.8.85:2377*

* + - * 1. Deploy Org4 services

**Come back to Manager Node terminal**

*./scripts/network/deploy\_services\_org4.sh*

After doing this step, Org4 related services should show up on Manager node (*docker service ls*) and Org4 related containers should be up on Org4 node (*docker ps*).

### Join Org4 to the Channel

**Run below commands from Org4 node**

1. Login to **Org4** CLI container

*docker exec -it hlf\_services\_cli.1.pg1lfx232ocy2vc3jiknz8m0j bash*

1. Export two environment variables

*export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer****1****.example.com/msp/tlscacerts/tlsca.example.com-cert.pem && export CHANNEL\_NAME=mychannel*

1. Get genesis.block of mychannel

*peer channel fetch 0 mychannel.block -o orderer****1****.example.com:7050 -c $CHANNEL\_NAME --tls --cafile $ORDERER\_CA*

1. Finally, join peer**0**.org4.example.com to the channel

*export CORE\_PEER\_LOCALMSPID="Org4MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051*

*peer channel join -b mychannel.block*

After this step, ledger has to be copied to couchdb0\_org4 instance (which is attached to peer0.org4.example.com). Just cross verify. If you are able to see existing documents in ‘mychannel\_couchdb\_java\_simple’ database, then everything went well.

In browser,<http://3.23.127.214:11984/_utils/#database/mychannel_couchdb_java_simple/_all_docs>

1. Finally, join peer**1**.org4.example.com also to the channel

*export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer1.org4.example.com/tls/ca.crt && export CORE\_PEER\_ADDRESS=peer1.org4.example.com:7051;*

*peer channel join -b mychannel.block*

Check whether couchdb1\_org4 instance is updated with records are not.

In browser,[http://3.23.127.214:1**2**984/\_utils/#database/mychannel\_couchdb\_java\_simple/\_all\_docs](http://3.23.127.214:12984/_utils/#database/mychannel_couchdb_java_simple/_all_docs)

### Upgrade and Invoke Chaincode on Org4’s Peers

* + - * 1. ~~From Org4 terminal (not from CLI container),copy ‘HLF-Chaincode-Java-CouchDB-Rich-Queries’ contents within Org4 CLI container (this has to be done by ./copy\_org4\_crypto\_and\_scp\_other\_nodes.sh script but not happening. TODO: have to check)~~

*~~docker cp /home/ubuntu/HLF-Multi-Host-Docker-Swarm/chaincodes/HLF-Chaincode-Java-CouchDB-Rich-Queries hlf\_services\_cli.1.pg1lfx232ocy2vc3jiknz8m0j:/opt/gopath/src/github.com/chaincode~~*

* + - * 1. From Org4 terminal (not from CLI container), copy CC package file into Org4 CLI container

*docker cp couchdb\_java\_simple\_v1.pak hlf\_services\_cli\_cli.1.q8oqwsna2ido05xispy6jyuev:/opt/gopath/src/github.com/hyperledger/fabric/peer*

* + - * 1. Continue to execute below commands from **Org4** CLI container. If you logged into container then you need to export below environment variables.

*export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer****1****.example.com/msp/tlscacerts/tlsca.example.com-cert.pem && export CHANNEL\_NAME=mychannel*

* + - * 1. Upgrade CC on peer**0**.org4.example.com

*export CORE\_PEER\_LOCALMSPID="Org4MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051*

Now, upgrade CC

*peer chaincode install couchdb\_java\_simple\_v1.pak*

*~~peer chaincode~~* ***~~install~~*** *~~-l java -n couchdb\_java\_simple -v v3 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

Just cross verify whether CC is installed properly or not

*peer chaincode list --installed*

**Note**: If you want to install CC on 2nd peer of Org4 then do below 3rd step. Otherwise, it is not required. Since, in this example I haven’t installed CC on 2nd peers of Orgs (ex: peer1.org1.example.com), I will not install on peer1.org4.example.com. Same applicable for other Orgs as well.

* + - * 1. ~~Upgrade CC on peer~~**~~1~~**~~.org4.example.com~~

*~~export CORE\_PEER\_LOCALMSPID="Org4MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer1.org4.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp;export CORE\_PEER\_ADDRESS=peer1.org4.example.com:7051~~*

~~Now, upgrade CC~~

*~~peer chaincode install couchdb\_java\_simple\_v1.pak~~*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

### Upgrade and Invoke Chaincode on other 3 Org’s Peers

Login into Org1 CLI container (you must have already logged in to this CLI from other terminal)

*docker exec -it hlf\_services\_cli.1.kr38ownvvy6unqd5uf0wmgjxt bash*

*export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer****1****.example.com/msp/tlscacerts/tlsca.example.com-cert.pem && export CHANNEL\_NAME=mychannel*

* + - * 1. Upgrade CC on peer0.**org1**.example.com

*export CORE\_PEER\_LOCALMSPID="Org1MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051*

Now, upgrade CC

*peer chaincode install couchdb\_java\_simple\_v1.pak*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

If you have installed CC on 2nd peer of Org1 then do below 2nd step. Otherwise, it is not required. Since, in this example I haven’t installed CC on peer1.org1.example.com, I will not do this step. Same applicable for other Orgs as well.

* + - * 1. ~~Upgrade CC on peer~~**~~1~~**~~.org1.example.com~~

*~~export CORE\_PEER\_LOCALMSPID="Org1MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp;export CORE\_PEER\_ADDRESS=peer1.org1.example.com:7051~~*

~~Now, upgrade CC~~

*~~peer chaincode install couchdb\_java\_simple\_v1.pak~~*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

* + - * 1. Upgrade CC on peer0.**org2**.example.com

*export CORE\_PEER\_LOCALMSPID="Org2MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051*

Now, upgrade CC

*peer chaincode install couchdb\_java\_simple\_v1.pak*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

* + - * 1. ~~Upgrade CC on peer~~**~~1~~**~~.org2.example.com~~

*~~export CORE\_PEER\_LOCALMSPID="Org2MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer1.org2.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp;export CORE\_PEER\_ADDRESS=peer1.org2.example.com:7051~~*

~~Now, upgrade CC~~

*~~peer chaincode install couchdb\_java\_simple\_v1.pak~~*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

* + - * 1. Upgrade CC on peer0.**org3**.example.com

*export CORE\_PEER\_LOCALMSPID="Org3MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051*

Now, upgrade CC

*peer chaincode install couchdb\_java\_simple\_v1.pak*

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

* + - * 1. ~~Upgrade CC on peer~~**~~1~~**~~.org3.example.com~~

*~~export CORE\_PEER\_LOCALMSPID="Org3MSP";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer1.org3.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp;export CORE\_PEER\_ADDRESS=peer1.org3.example.com:7051~~*

~~Now, upgrade CC~~

*~~peer chaincode install -l java -n couchdb\_java\_simple -v v1 -p /opt/gopath/src/github.com/chaincode/HLF-Chaincode-Java-CouchDB-Rich-Queries~~*

* + - * 1. **Important step**: Upgrade CC to include Org4 in endorsement policy

*export CORE\_PEER\_LOCALMSPID="****Org1MSP****";export CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt;export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp;export CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051*

Now, upgrade CC **(this is the step actually instantiate new version CC)**

*peer chaincode* ***upgrade*** *-o orderer1.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C $CHANNEL\_NAME -l java -n couchdb\_java\_simple -v v1 -c '{"Args":["init", "34", "org4\_name", "org4\_dept", "55", "org4\_loc"]}' -P "OR ('Org1MSP.peer','Org2MSP.peer','Org3MSP.peer',****'Org4MSP.peer'****)"*

### Just check whether everything is went well or not

Login into Org4 CLI container

*docker exec -it hlf\_services\_cli.1.kr38ownvvy6unqd5uf0wmgjxt bash*

*export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer****1****.example.com/msp/tlscacerts/tlsca.example.com-cert.pem && export CHANNEL\_NAME=mychannel*

Just query the CC to check whether upgraded properly or not

*peer chaincode invoke -C $CHANNEL\_NAME -n couchdb\_java\_simple -c '{"Args":["queryEmployee","1"]}'*

Just try adding to another employee to double sure everything working

*peer chaincode invoke -C $CHANNEL\_NAME -n couchdb\_java\_simple -c '{"Args":["addEmployee", "30", "new\_org4\_name", "new\_org4\_dept", "55", "new\_org4\_loc"]}'*

**The endgame is to form a delta transaction object represented in protobuf binary format and then acquire the requisite number of admin signatures such that the channel configuration update transaction fulfills the channel’s modification policy.**

### Updating the Channel Config to include an Org4 Anchor Peer (Optional)

Continue from same Org4CLI container session only

Fetch the latest channel configuration

*peer channel fetch config config\_block.pb -o orderer1.example.com:7050 -c $CHANNEL\_NAME --tls --cafile $ORDERER\_CA*

Convert config block to json

*configtxlator proto\_decode --input config\_block.pb --type common.Block | jq .data.data[0].payload.data.config > config.json*

Update the configuration JSON with the Org3 anchor peer we want to add

*jq '.channel\_group.groups.Application.groups.Org4MSP.values += {"AnchorPeers":{"mod\_policy": "Admins","value":{"anchor\_peers": [{"host": "peer0.org4.example.com","port": 7051}]},"version": "0"}}' config.json > modified\_anchor\_config.json*

Translate config.json back into protobuf format as config.pb

*configtxlator proto\_encode --input config.json --type common.Config --output config.pb*

Translate the modified\_anchor\_config.json into protobuf format as modified\_anchor\_config.pb

*configtxlator proto\_encode --input modified\_anchor\_config.json --type common.Config --output modified\_anchor\_config.pb*

Calculate the delta between the two protobuf formatted configurations

*configtxlator compute\_update --channel\_id $CHANNEL\_NAME --original config.pb --updated modified\_anchor\_config.pb --output anchor\_update.pb*

Convert anchor\_update.pb into anchor\_update.json

*configtxlator proto\_decode --input anchor\_update.pb --type common.ConfigUpdate | jq . > anchor\_update.json*

Next we will wrap the update in an envelope message, restoring the previously stripped away header, outputting it to anchor\_update\_in\_envelope.json

*echo '{"payload":{"header":{"channel\_header":{"channel\_id":"'$CHANNEL\_NAME'", "type":2}},"data":{"config\_update":'$(cat anchor\_update.json)'}}}' | jq . > anchor\_update\_in\_envelope.json*

Convert it to a protobuf

*configtxlator proto\_encode --input anchor\_update\_in\_envelope.json --type common.Envelope --output anchor\_update\_in\_envelope.pb*

Finally, apply Anchor peer update

*peer channel update -f anchor\_update\_in\_envelope.pb -c $CHANNEL\_NAME -o orderer1.example.com:7050 --tls --cafile $ORDERER\_CA*

Just cross verify whether above changes are applied property or not. Check updates in peer0.org1 logs

*docker logs hlf\_peer\_peer0\_org1.1.e01r3ysxn3aszwblv5gz98hqd 2>&1 | grep "Learning about the configured anchor peers of Org4MSP for channel mychannel"*

2020-05-04 07:02:01.175 UTC [gossip.gossip] learnAnchorPeers -> INFO 19e Learning about the configured anchor peers of **Org4MSP** for channel mychannel : [{**peer0.org4.example.com** 7051}]