Adding a new organization to the existing network

In this lab guide, we will try adding a new organization org3 to the existing network.

Note: Make sure you have a working test network (Fabric-network) with all the certificates generated, and the containers are up and running. Also, ensure that the chaincode is deployed successfully, and three command terminals(host, peer0_Org1 and peer0_Org2) are opened with the necessary environment variables set. Otherwise, follow the steps mentioned in the Lab Manual - Setting up the Hyperledger Fabric network.

Before we start, let's brief the steps involved:

- Generating crypto materials for the new Org that is to be added.
- Creating docker-compose files for the new org, to launch the new org components such as peer, couchdb and ca.
- Fetching the channel config and updating it by adding the new org.
- Sign and submit config update
- Joining org3 to the channel.
- Anchor peer update
- Installing and invoking chaincode to the org3 peer.

Step1. Generating crypto materials for the new org.

Inside the docker folder create a file and name it **docker-compose-ca-org3.yaml** . Add the following. (Refer: fabric-samples/test-network/addOrg3/compose/compose-ca-org3.yaml)

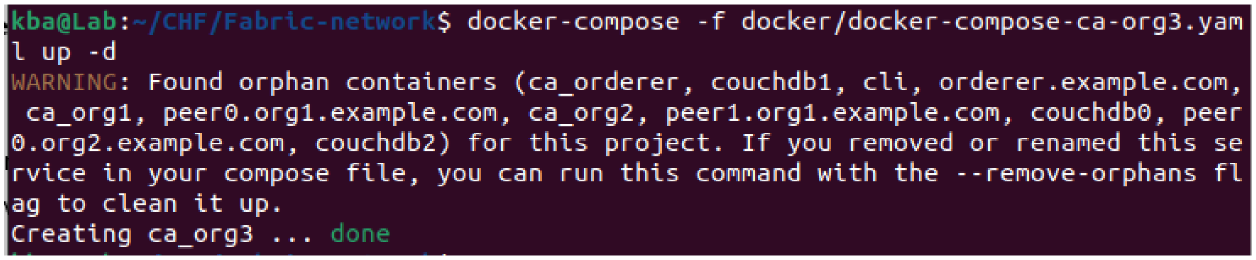
```
version: '3.7'

networks:
  test:
    name: fabric_test
```

```
services:
  ca_org3:
    image: hyperledger/fabric-ca:latest
    labels:
      service: hyperledger-fabric
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-org3
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_PORT=11054
      - FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS=0.0.0.0:11055
    ports:
      - "11054:11054"
      - "11055:11055"
    command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
    volumes:
      - ../organizations/fabric-ca/org3:/etc/hyperledger/fabric-ca-server
    container_name: ca_org3
    networks:
      - test
```

To run the docker-compose file, you can use the following command. Execute the following commands from within the **Fabric-network** folder.

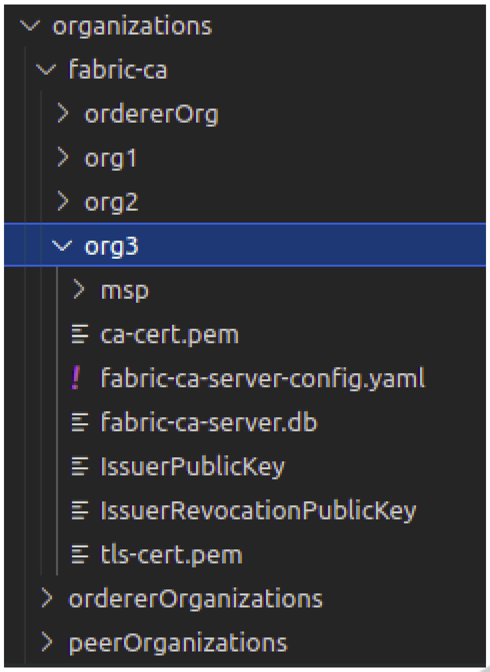
Execute the command in **host** terminal

docker-compose -f docker/docker-compose-ca-org3.yaml up -d



```
kba@Lab:~/CHF/Fabric-network$ docker-compose -f docker/docker-compose-ca-org3.yam
l up -d
WARNING: Found orphan containers (ca_orderer, couchdb1, cli, orderer.example.com,
 ca_org1, peer0.org1.example.com, ca_org2, peer1.org1.example.com, couchdb0, peer
0.org2.example.com, couchdb2) for this project. If you removed or renamed this se
rvice in your compose file, you can run this command with the --remove-orphans fl
ag to clean it up.
Creating ca_org3 ... done
```

Now a container for ca_org3 will be started, and all the required certificates and other related files will be generated under **organizations/fabric-ca/org3** folder.



```
organizations
  fabric-ca
    ordererOrg
    org1
    org2
    org3
      msp
      ca-cert.pem
      fabric-ca-server-config.yaml
      fabric-ca-server.db
      IssuerPublicKey
      IssuerRevocationPublicKey
      tls-cert.pem
  ordererOrganizations
  peerOrganizations
```

If the org3 folder is locked, then unlock it using:

Execute the command in **host** terminal

sudo chmod -R 777 organizations/fabric-ca/org3

Let's write a script file for registering and enrolling the various identities for Org3 and organizing the certificates in particular folders. Create a file **registerEnrollOrg3.sh** in Fabric-network folder and add the following (Refer: fabric-samples/test-network/addOrg3/fabric-ca/registerEnroll.sh)

```
#!/bin/bash

function createOrg3() {
  echo "Enrolling the CA admin"
  mkdir -p organizations/peerOrganizations/org3.example.com/

  export
FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org3.example.c
om/
```

```
  set -x
  fabric-ca-client enroll -u https://admin:adminpw@localhost:11054 --caname
ca-org3 --tls.certfiles "${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  echo 'NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: cacerts/localhost-11054-ca-org3.pem
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: cacerts/localhost-11054-ca-org3.pem
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: cacerts/localhost-11054-ca-org3.pem
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: cacerts/localhost-11054-ca-org3.pem
    OrganizationalUnitIdentifier: orderer' >
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/config.yaml"

  # Since the CA serves as both the organization CA and TLS CA, copy the
org's root cert that was generated by CA startup into the org level ca and
tlsca directories

  # Copy org3's CA cert to org3's /msp/tlscacerts directory (for use in the
channel MSP definition)
  mkdir -p
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/tlscacerts"
  cp "${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/tlscacerts/ca.
crt"

  # Copy org3's CA cert to org3's /tlsca directory (for use by clients)
  mkdir -p "${PWD}/organizations/peerOrganizations/org3.example.com/tlsca"
  cp "${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
"${PWD}/organizations/peerOrganizations/org3.example.com/tlsca/tlsca.org3.e
xample.com-cert.pem"

  # Copy org3's CA cert to org3's /ca directory (for use by clients)
  mkdir -p "${PWD}/organizations/peerOrganizations/org3.example.com/ca"
  cp "${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
```

```
"${PWD}/organizations/peerOrganizations/org3.example.com/ca/ca.org3.example
.com-cert.pem"

  echo "Registering peer0"
  set -x
  fabric-ca-client register --caname ca-org3 --id.name peer0 --id.secret
peer0pw --id.type peer --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  echo "Registering user"
  set -x
  fabric-ca-client register --caname ca-org3 --id.name user1 --id.secret
user1pw --id.type client --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  echo "Registering the org admin"
  set -x
  fabric-ca-client register --caname ca-org3 --id.name org3admin
--id.secret org3adminpw --id.type admin --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  echo "Generating the peer0 msp"
  set -x
  fabric-ca-client enroll -u https://peer0:peer0pw@localhost:11054 --caname
ca-org3 -M
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/msp" --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/config.yaml"
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/msp/config.yaml"

  echo "Generating the peer0-tls certificates, use --csr.hosts to specify
Subject Alternative Names"
  set -x
  fabric-ca-client enroll -u https://peer0:peer0pw@localhost:11054 --caname
```

```
ca-org3 -M
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls" --enrollment.profile tls --csr.hosts peer0.org3.example.com
--csr.hosts localhost --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  # Copy the tls CA cert, server cert, server keystore to well known file
names in the peer's tls directory that are referenced by peer startup
config
  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/tlscacerts/"*
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/ca.crt"
  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/signcerts/"*
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/server.crt"
  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/keystore/"*
"${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.e
xample.com/tls/server.key"

  echo "Generating the user msp"
  set -x
  fabric-ca-client enroll -u https://user1:user1pw@localhost:11054 --caname
ca-org3 -M
"${PWD}/organizations/peerOrganizations/org3.example.com/users/User1@org3.e
xample.com/msp" --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/config.yaml"
"${PWD}/organizations/peerOrganizations/org3.example.com/users/User1@org3.e
xample.com/msp/config.yaml"

  echo "Generating the org admin msp"
  set -x
  fabric-ca-client enroll -u https://org3admin:org3adminpw@localhost:11054
--caname ca-org3 -M
"${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.e
xample.com/msp" --tls.certfiles
"${PWD}/organizations/fabric-ca/org3/ca-cert.pem"
  { set +x; } 2>/dev/null

  cp
"${PWD}/organizations/peerOrganizations/org3.example.com/msp/config.yaml"
"${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.e
xample.com/msp/config.yaml"
}

createOrg3
```

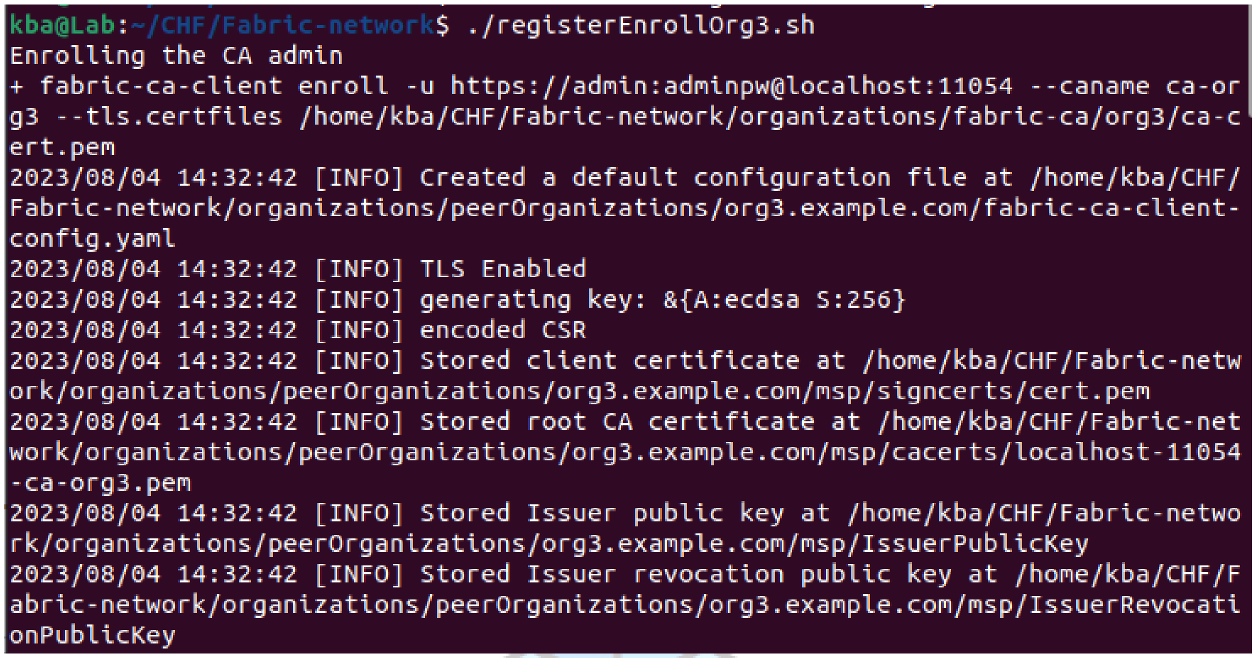
Now run the **registerEnrollOrg3.sh** file. First, make the script file executable using the command:

Execute the command in **host** terminal

chmod +x registerEnrollOrg3.sh

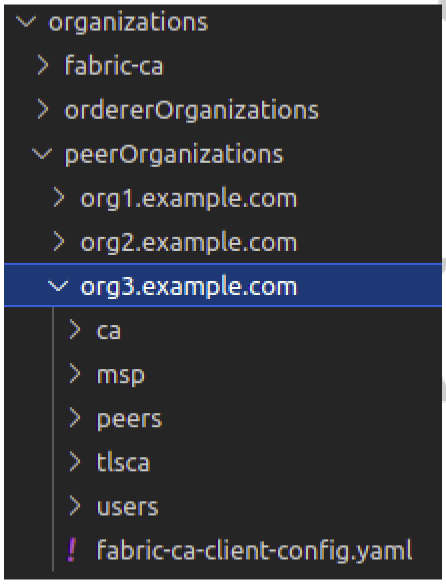
Run the script file using the command:

./registerEnrollOrg3.sh



```
kba@Lab:~/CHF/Fabric-network$ ./registerEnrollOrg3.sh
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:11054 --caname ca-or
g3 --tls.certfiles /home/kba/CHF/Fabric-network/organizations/fabric-ca/org3/ca-c
ert.pem
2023/08/04 14:32:42 [INFO] Created a default configuration file at /home/kba/CHF/
Fabric-network/organizations/peerOrganizations/org3.example.com/fabric-ca-client-
config.yaml
2023/08/04 14:32:42 [INFO] TLS Enabled
2023/08/04 14:32:42 [INFO] generating key: &{A:ecdsa S:256}
2023/08/04 14:32:42 [INFO] encoded CSR
2023/08/04 14:32:42 [INFO] Stored client certificate at /home/kba/CHF/Fabric-netw
ork/organizations/peerOrganizations/org3.example.com/msp/signcerts/cert.pem
2023/08/04 14:32:42 [INFO] Stored root CA certificate at /home/kba/CHF/Fabric-net
work/organizations/peerOrganizations/org3.example.com/msp/cacerts/localhost-11054
-ca-org3.pem
2023/08/04 14:32:42 [INFO] Stored Issuer public key at /home/kba/CHF/Fabric-netwo
rk/organizations/peerOrganizations/org3.example.com/msp/IssuerPublicKey
2023/08/04 14:32:42 [INFO] Stored Issuer revocation public key at /home/kba/CHF/F
abric-network/organizations/peerOrganizations/org3.example.com/msp/IssuerRevocati
onPublicKey
```

It creates the respective folders and certificates within the **organizations** folder.



Create a folder **configOrg3** within the Fabric-network folder.

Execute the command in **host** terminal

mkdir configOrg3

Create a **configtx.yaml** file under **configOrg3** folder and add the following. (Refer: fabric-samples/test-network/addOrg3/configtx.yaml)

```
---
Organizations:
  - &Org3
    Name: Org3MSP

    ID: Org3MSP

    MSPDir: ../organizations/peerOrganizations/org3.example.com/msp

    Policies:
      Readers:
        Type: Signature
        Rule: "OR('Org3MSP.admin', 'Org3MSP.peer', 'Org3MSP.client')"
      Writers:
        Type: Signature
        Rule: "OR('Org3MSP.admin', 'Org3MSP.client')"
      Admins:
        Type: Signature
        Rule: "OR('Org3MSP.admin')"
      Endorsement:
        Type: Signature
        Rule: "OR('Org3MSP.peer')"
```

We can use the configtxgen tool to print out the Org3 organization definition. First, set the FABRIC_CFG_PATH environment variable to the directory's path that contains the configtx.yaml file. For that, execute the following command from within the Fabric-network folder:

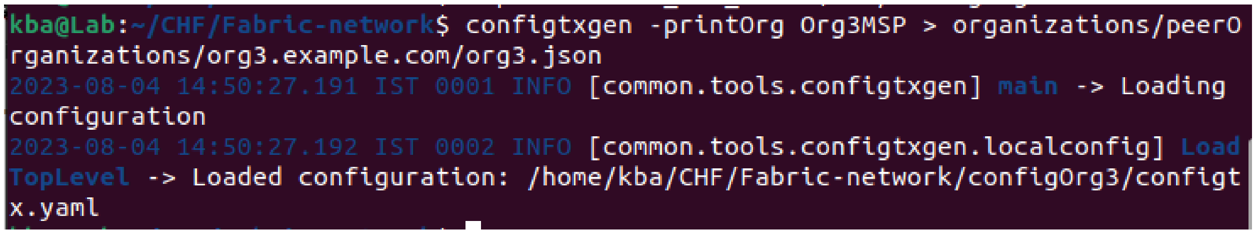
Execute the command in **host** terminal

export FABRIC_CFG_PATH=$PWD/configOrg3

Add the Org3 configuration to a json file by executing the following command:

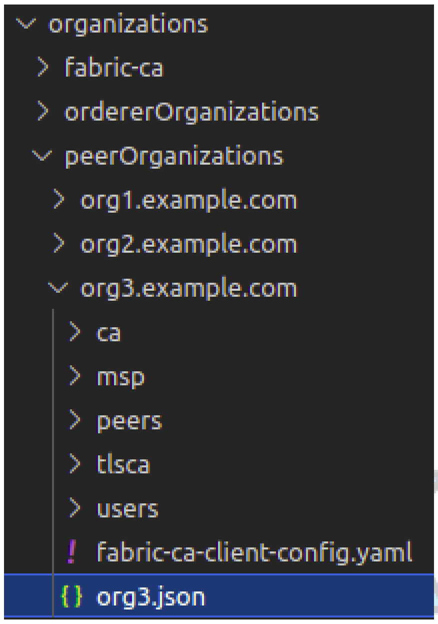
Execute the command in **host** terminal

configtxgen -printOrg Org3MSP > organizations/peerOrganizations/org3.example.com/org3.json



```
kba@Lab:~/CHF/Fabric-network$ configtxgen -printOrg Org3MSP > organizations/peerO
rganizations/org3.example.com/org3.json
2023-08-04 14:50:27.191 IST 0001 INFO [common.tools.configtxgen] main -> Loading
configuration
2023-08-04 14:50:27.192 IST 0002 INFO [common.tools.configtxgen.localconfig] Load
TopLevel -> Loaded configuration: /home/kba/CHF/Fabric-network/configOrg3/configt
x.yaml
```

It creates the **org3.json** file within the org3.example.com folder.



```
organizations
  fabric-ca
  ordererOrganizations
  peerOrganizations
    org1.example.com
    org2.example.com
    org3.example.com
      ca
      msp
      peers
      tlsca
      users
      fabric-ca-client-config.yaml
      org3.json
```

Step2. Creating docker-compose files for the new org.

Inside the docker folder, create a file **docker-compose-org3.yaml** and add the following. (Refer: fabric-samples/test-network/addOrg3/compose/compose-org3.yaml, fabric-samples/test-network/addOrg3/compose/compose-couch-org3.yaml)

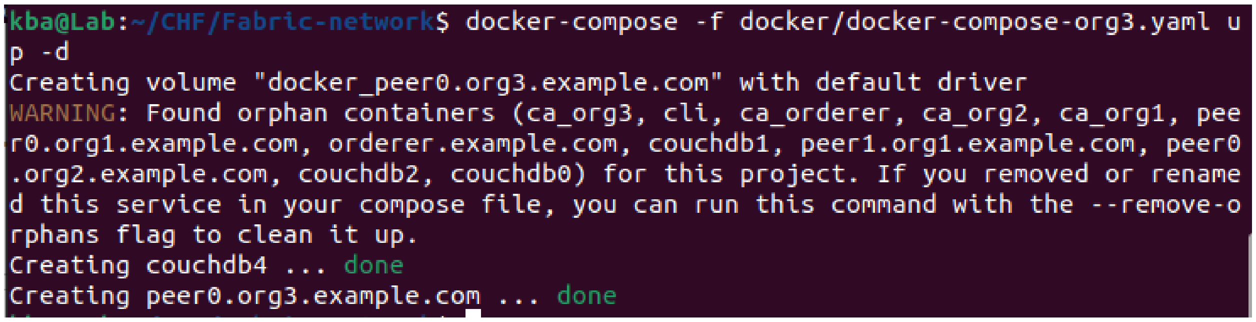
```
version: '3.7'

volumes:
  peer0.org3.example.com:
```

```
networks:
  test:
    name: fabric_test

services:

  couchdb4:
    container_name: couchdb4
    image: couchdb:3.3.2
    labels:
      service: hyperledger-fabric
    environment:
      - COUCHDB_USER=admin
      - COUCHDB_PASSWORD=adminpw
    ports:
      - "9984:5984"
    networks:
      - test

  peer0.org3.example.com:
    container_name: peer0.org3.example.com
    image: hyperledger/fabric-peer:latest
    labels:
      service: hyperledger-fabric
    environment:
      # - FABRIC_CFG_PATH=/etc/hyperledger/peercfg
      #Generic peer variables
      - FABRIC_LOGGING_SPEC=INFO
      #- FABRIC_LOGGING_SPEC=DEBUG
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=fabric_test
      - CORE_PEER_TLS_ENABLED=true
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
      - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
      - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
      # Peer specific variables
      - CORE_PEER_ID=peer0.org3.example.com
      - CORE_PEER_ADDRESS=peer0.org3.example.com:11051
      - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/fabric/msp
      - CORE_PEER_LISTENADDRESS=0.0.0.0:11051
      - CORE_PEER_CHAINCODEADDRESS=peer0.org3.example.com:11052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:11052
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org3.example.com:11051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org3.example.com:11051
      - CORE_PEER_LOCALMSPID=Org3MSP
      - CORE_OPERATIONS_LISTENADDRESS=peer0.org3.example.com:9446
      - CORE_METRICS_PROVIDER=prometheus
      - CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG={"peername":"peer0org3"}
      - CORE_CHAINCODE_EXECUTETIMEOUT=300s
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb4:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
    depends_on:
      - couchdb4
    volumes:
      - /var/run/docker.sock:/host/var/run/docker.sock
      -
../organizations/peerOrganizations/org3.example.com/peers/peer0.org3.exampl
e.com:/etc/hyperledger/fabric
      - peer0.org3.example.com:/var/hyperledger/production
    working_dir: /root
    command: peer node start
    ports:
      - 11051:11051
      - 9446:9446
    networks:
      - test
```

Now run the docker-compose file. For that, execute the following command from the Fabric-network folder.

Execute the command in **host** terminal

docker-compose -f docker/docker-compose-org3.yaml up -d



```
kba@Lab:~/CHF/Fabric-network$ docker-compose -f docker/docker-compose-org3.yaml u
p -d
Creating volume "docker_peer0.org3.example.com" with default driver
WARNING: Found orphan containers (ca_org3, cli, ca_orderer, ca_org2, ca_org1, pee
r0.org1.example.com, orderer.example.com, couchdb1, peer1.org1.example.com, peer0
.org2.example.com, couchdb2, couchdb0) for this project. If you removed or rename
d this service in your compose file, you can run this command with the --remove-o
rphans flag to clean it up.
Creating couchdb4 ... done
Creating peer0.org3.example.com ... done
```

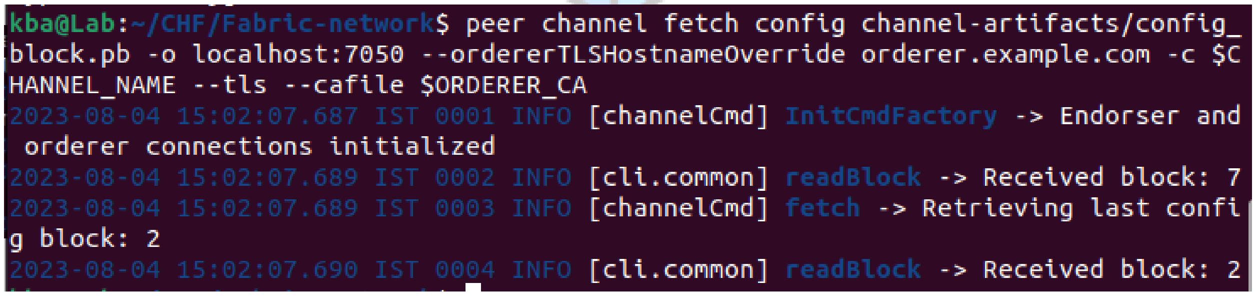
Created two more containers **couchdb4** and **peer0.org3.example.com** .

Step3. Fetching the channel configuration.

Now, we need to fetch the most recent config block for the channel. Since Org1 is a member of the channel, the Org1 admin has permission to fetch the channel config from the ordering service. So let's execute the **peer channel fetch** command from Org1.

Execute the command in **peer0_Org1** terminal

peer channel fetch config channel-artifacts/config_block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL_NAME --tls --cafile $ORDERER_CA



```
kba@Lab:~/CHF/Fabric-network$ peer channel fetch config channel-artifacts/config_
block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $C
HANNEL_NAME --tls --cafile $ORDERER_CA
2023-08-04 15:02:07.687 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and
 orderer connections initialized
2023-08-04 15:02:07.689 IST 0002 INFO [cli.common] readBlock -> Received block: 7
2023-08-04 15:02:07.689 IST 0003 INFO [channelCmd] fetch -> Retrieving last confi
g block: 2
2023-08-04 15:02:07.690 IST 0004 INFO [cli.common] readBlock -> Received block: 2
```

This command saves the binary protobuf channel configuration block to **config_block.pb**.

Now change the directory to 'channel-artifacts' folder using

cd channel-artifacts

Now, we need to convert the Configuration to JSON and trim it down using the **configtxlator** tool and **jq** tool. Execute the following commands.

Execute the command in **peer0_Org1** terminal

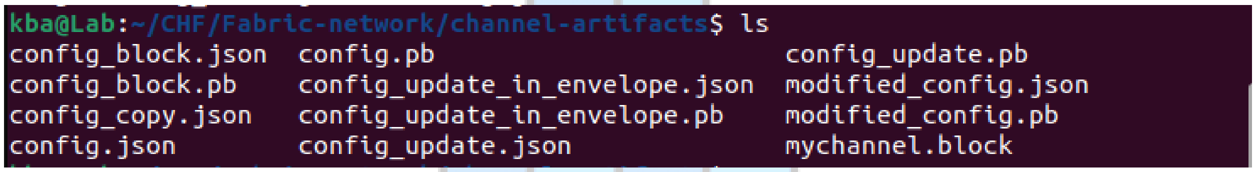
configtxlator proto_decode --input config_block.pb --type common.Block --output config_block.json

jq ".data.data[0].payload.data.config" config_block.json > config.json

By executing the **ls** command, we can see the generated files.

Execute the command in **peer0_Org1** terminal

ls



```
kba@Lab:~/CHF/Fabric-network/channel-artifacts$ ls
config_block.json  config.pb                         config_update.pb
config_block.pb    config_update_in_envelope.json    modified_config.json
config_copy.json   config_update_in_envelope.pb      modified_config.pb
config.json        config_update.json                mychannel.block
```

Now, add the Org3 configuration definition, **org3.json**, to the channel's application groups field using the **jq** tool.

Execute the command in **peer0_Org1** terminal

jq -s '.[0] * {"channel_group":{"groups":{"Application":{"groups": {"Org3MSP":.[1]}}}}}' config.json ../organizations/peerOrganizations/org3.example.com/org3.json > modified_config.json

Convert **config.json** back into a protobuf by using **configtxlator** tool. Executing the following command.

Execute the command in **peer0_Org1** terminal

configtxlator proto_encode --input config.json --type common.Config --output config.pb

Next, encode **modified_config.json** to **modified_config.pb**:

Execute the command in **peer0_Org1** terminal

```
configtxlator proto_encode --input modified_config.json --type common.Config
--output modified_config.pb
```

Now use **configtxlator** to calculate the delta between these two config protobufs. This command will output a new protobuf binary named **org3_update.pb**:

Execute the command in **peer0_Org1** terminal

```
configtxlator compute_update --channel_id $CHANNEL_NAME --original
config.pb --updated modified_config.pb --output org3_update.pb
```

This new proto – org3_update.pb – contains the Org3 definitions and high level pointers to the Org1 and Org2 material.

Let’s decode this object into editable JSON format:

Execute the command in **peer0_Org1** terminal

```
configtxlator proto_decode --input org3_update.pb --type
common.ConfigUpdate --output org3_update.json
```

Now, we have a decoded update file – org3_update.json – that we need to wrap in an envelope message. This step will give us back the header field that we stripped away earlier.

Execute the command in **peer0_Org1** terminal

```
echo
'{"payload":{"header":{"channel_header":{"channel_id":"'$CHANNEL_NAME'",
"type":2}},"data":{"config_update":'$(cat org3_update.json)'}}}' | jq . >
org3_update_in_envelope.json
```

Using the configtxlator tool, convert it into the fully fledged protobuf format that Fabric requires.

Execute the command in **peer0_Org1** terminal

configtxlator proto_encode --input org3_update_in_envelope.json --type common.Envelope --output org3_update_in_envelope.pb

Step4. Sign and submit config update.

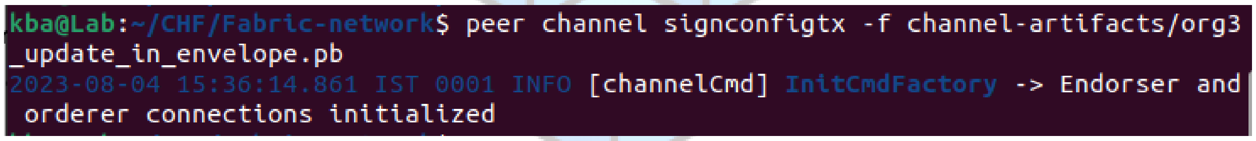
First come out of the 'channel-artifacts' folder using **cd ..**

cd ..

We need signatures from the requisite Admin users before the config can be written to the ledger. First, let's sign this update proto as Org1.

Execute the command in **peer0_Org1** terminal

peer channel signconfigtx -f channel-artifacts/org3_update_in_envelope.pb

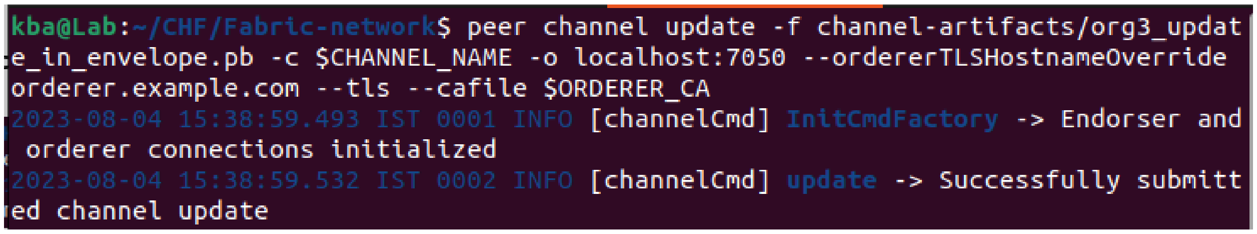


```
kba@Lab:~/CHF/Fabric-network$ peer channel signconfigtx -f channel-artifacts/org3
_update_in_envelope.pb
2023-08-04 15:36:14.861 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and
 orderer connections initialized
```

Now we have to add the signature of **Org2**. So issue the **peer channel update** command. The **Org2 Admin** signature will be attached to this call so there is no need to manually sign the protobuf a second time:

#####Execute the command in **peer0_Org2** terminal#####

peer channel update -f channel-artifacts/org3_update_in_envelope.pb -c $CHANNEL_NAME -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA



```
kba@Lab:~/CHF/Fabric-network$ peer channel update -f channel-artifacts/org3_update_in_envelope.pb -c $CHANNEL_NAME -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA
2023-08-04 15:38:59.493 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-08-04 15:38:59.532 IST 0002 INFO [channelCmd] update -> Successfully submitted channel update
```

Step5. Joining org3 to the channel.

Now, the channel configuration has been updated to include our new organization. So we can join the Org3 peer to the channel. So open another command terminal and update the environment variables in the Fabric CLI container accordingly. Let's mention this terminal as **peer0_Org3 terminal**, because we will set the environment variables that refer to peer0.org3.example.com .

Note: Hereafter, if any commands need to be executed in this terminal, we will mention them as **peer0_Org3 terminal**.

Here let's set the environment variables corresponding to peer0.org3.example.com. Execute the following commands.

#####Execute the command in **peer0_Org3** terminal#####

export FABRIC_CFG_PATH=./peercfg

export CHANNEL_NAME=mychannel

export CORE_PEER_TLS_ENABLED=true

export CORE_PEER_LOCALMSPID=Org3MSP

export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt

export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp

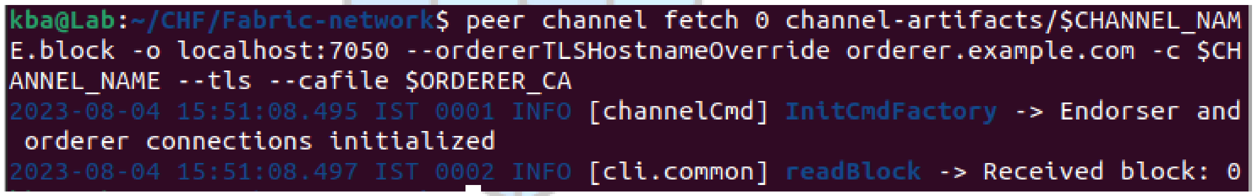
export CORE_PEER_ADDRESS=localhost:11051

export ORDERER_CA=${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

As a result of the successful channel update, the ordering service will verify that Org3 can pull the genesis block and join the channel. Use the **peer channel fetch** command to retrieve this block:

#####Execute the command in **peer0_Org3** terminal#####

peer channel fetch 0 channel-artifacts/$CHANNEL_NAME.block -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL_NAME --tls --cafile $ORDERER_CA



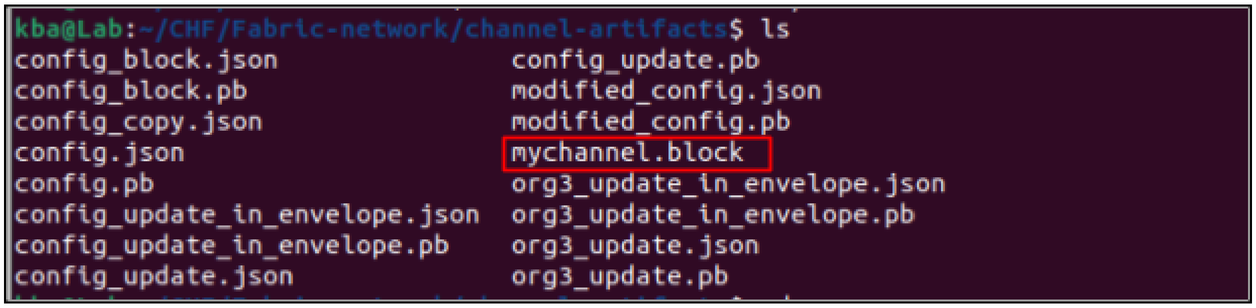
```
kba@Lab:~/CHF/Fabric-network$ peer channel fetch 0 channel-artifacts/$CHANNEL_NAM
E.block -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CH
ANNEL_NAME --tls --cafile $ORDERER_CA
2023-08-04 15:51:08.495 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and
 orderer connections initialized
2023-08-04 15:51:08.497 IST 0002 INFO [cli.common] readBlock -> Received block: 0
```

If successful, the command returned the genesis block to a file named **mychannel.block** inside the channel-artifacts folder. Confirm it by giving the command **cd channel-artifacts** and **ls.**

#####Execute the command in **peer0_Org3** terminal#####

cd channel-artifacts

ls


```
kba@Lab:~/CHF/Fabric-network/channel-artifacts$ ls
config_block.json                 config_update.pb
config_block.pb                   modified_config.json
config_copy.json                  modified_config.pb
config.json                       mychannel.block
config.pb                         org3_update_in_envelope.json
config_update_in_envelope.json    org3_update_in_envelope.pb
config_update_in_envelope.pb      org3_update.json
config_update.json                org3_update.pb
```

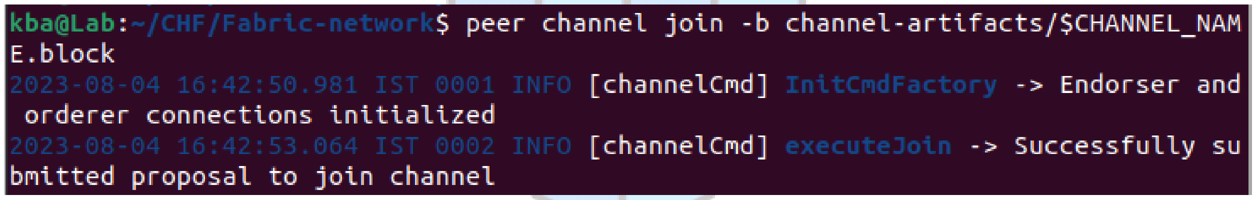
Now come back to Fabric-network folder

cd ..

Now use this block to join the peer to the channel.

#####Execute the command in **peer0_Org3** terminal#####

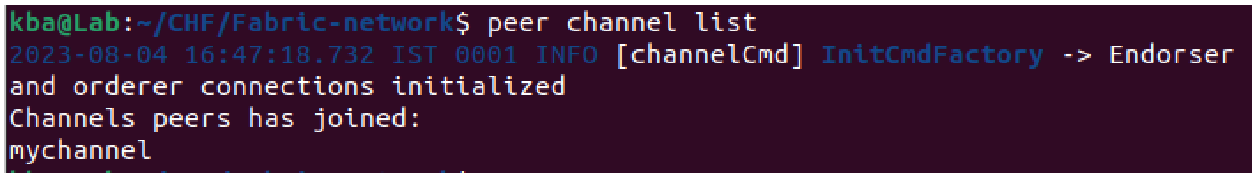
peer channel join -b channel-artifacts/$CHANNEL_NAME.block


```
kba@Lab:~/CHF/Fabric-network$ peer channel join -b channel-artifacts/$CHANNEL_NAM
E.block
2023-08-04 16:42:50.981 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and
 orderer connections initialized
2023-08-04 16:42:53.064 IST 0002 INFO [channelCmd] executeJoin -> Successfully su
bmitted proposal to join channel
```

Now, peer0.org3.example.com joined to mychannel. Confirm it by the peer channel list command.

#####Execute the command in **peer0_Org3** terminal#####

peer channel list

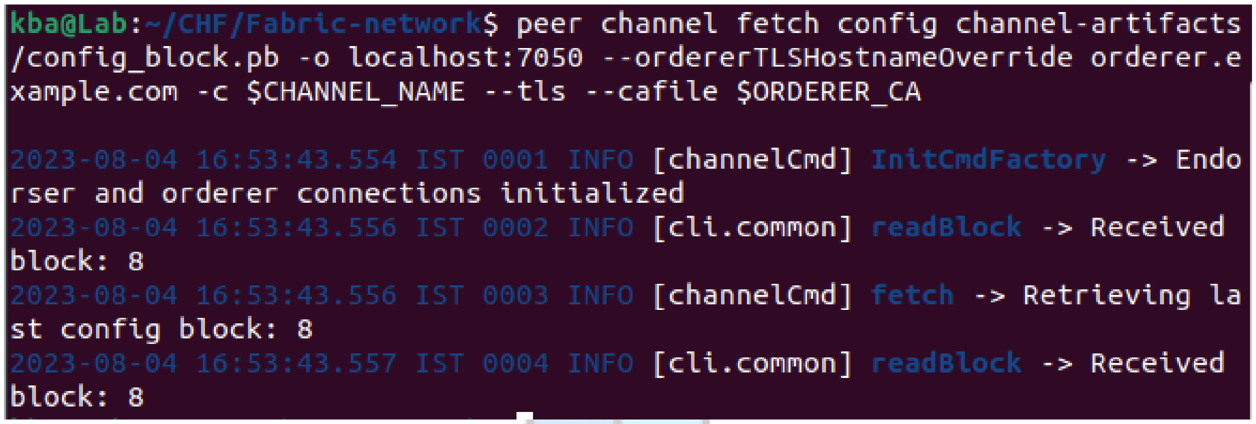

```
kba@Lab:~/CHF/Fabric-network$ peer channel list
2023-08-04 16:47:18.732 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser
and orderer connections initialized
Channels peers has joined:
mychannel
```

Step6. Anchor peer update

Execute the following commands for setting the anchor peer for org3.

#####Execute the command in **peer0_Org3** terminal#####

peer channel fetch config channel-artifacts/config_block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL_NAME --tls --cafile $ORDERER_CA



```
kba@Lab:~/CHF/Fabric-network$ peer channel fetch config channel-artifacts
/config_block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.e
xample.com -c $CHANNEL_NAME --tls --cafile $ORDERER_CA

2023-08-04 16:53:43.554 IST 0001 INFO [channelCmd] InitCmdFactory -> Endo
rser and orderer connections initialized
2023-08-04 16:53:43.556 IST 0002 INFO [cli.common] readBlock -> Received
block: 8
2023-08-04 16:53:43.556 IST 0003 INFO [channelCmd] fetch -> Retrieving la
st config block: 8
2023-08-04 16:53:43.557 IST 0004 INFO [cli.common] readBlock -> Received
block: 8
```

Now change directory to channel-artifacts folder

#####Execute the command in **peer0_Org3** terminal#####

cd channel-artifacts

Execute the following commands from **channel-artifacts** folder

#####Execute the command in **peer0_Org3** terminal#####

configtxlator proto_decode --input config_block.pb --type common.Block --output config_block.json

jq ".data.data[0].payload.data.config" config_block.json > config.json

jq '.channel_group.groups.Application.groups.Org3MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org3.example.com","port": 11051}]},"version": "0"}}' config.json > modified_anchor_config.json

configtxlator proto_encode --input config.json --type common.Config --output config.pb

configtxlator proto_encode --input modified_anchor_config.json --type common.Config --output modified_anchor_config.pb

configtxlator compute_update --channel_id $CHANNEL_NAME --original config.pb --updated modified_anchor_config.pb --output anchor_update.pb

configtxlator proto_decode --input anchor_update.pb --type common.ConfigUpdate --output anchor_update.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

configtxlator proto_encode --input anchor_update_in_envelope.json --type common.Envelope --output anchor_update_in_envelope.pb

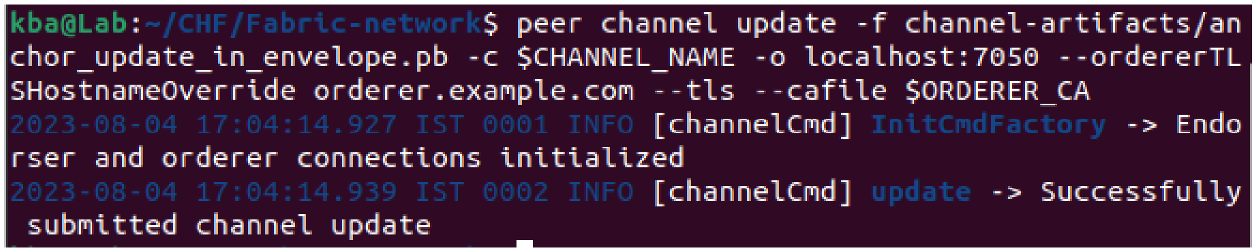
Now, come back to the Fabric-network folder

cd ..

And execute the following commands

#####Execute the command in **peer0_Org3** terminal#####

peer channel update -f channel-artifacts/anchor_update_in_envelope.pb -c $CHANNEL_NAME -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA



```
kba@Lab:~/CHF/Fabric-network$ peer channel update -f channel-artifacts/an
chor_update_in_envelope.pb -c $CHANNEL_NAME -o localhost:7050 --ordererTL
SHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA
2023-08-04 17:04:14.927 IST 0001 INFO [channelCmd] InitCmdFactory -> Endo
rser and orderer connections initialized
2023-08-04 17:04:14.939 IST 0002 INFO [channelCmd] update -> Successfully
 submitted channel update
```

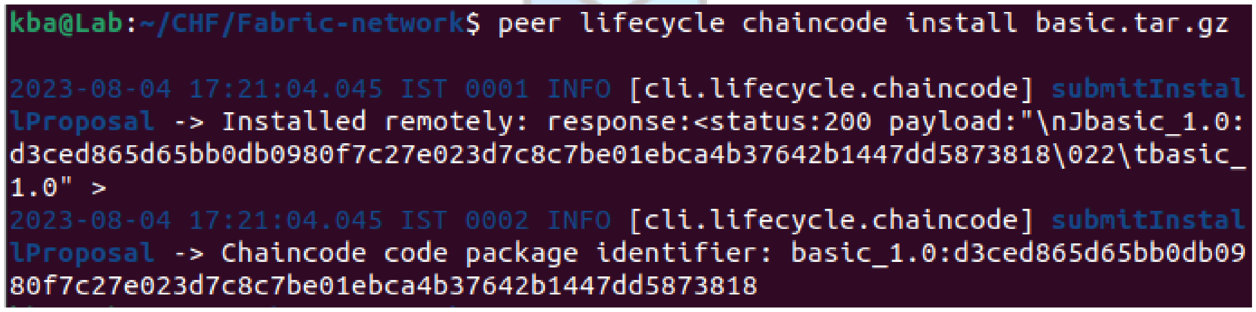
Now we have successfully updated the anchor peer. Let's do the next step,

Step7. Installing and Invoking chaincode to the org3 peer

Install the chaincode on peer0.org3.example.com

#####Execute the command in **peer0_Org3** terminal#####

peer lifecycle chaincode install basic.tar.gz

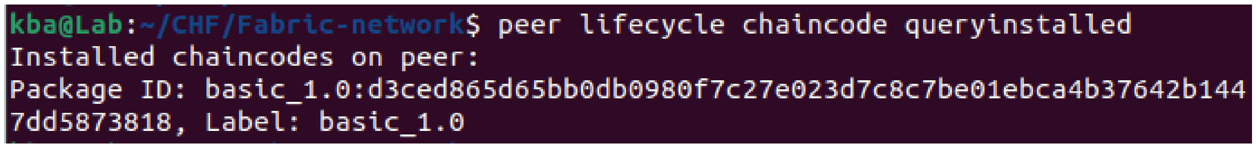


```
kba@Lab:~/CHF/Fabric-network$ peer lifecycle chaincode install basic.tar.gz

2023-08-04 17:21:04.045 IST 0001 INFO [cli.lifecycle.chaincode] submitInstal
lProposal -> Installed remotely: response:<status:200 payload:"\nJbasic_1.0:
d3ced865d65bb0db0980f7c27e023d7c8c7be01ebca4b37642b1447dd5873818\022\tbasic_
1.0" >
2023-08-04 17:21:04.045 IST 0002 INFO [cli.lifecycle.chaincode] submitInstal
lProposal -> Chaincode code package identifier: basic_1.0:d3ced865d65bb0db09
80f7c27e023d7c8c7be01ebca4b37642b1447dd5873818
```

It will return the package identifier(Package ID). You can query the peer and get a list of the installed chaincode packages using the following command.

#####Execute the command in **peer0_Org3** terminal#####

peer lifecycle chaincode queryinstalled



```
kba@Lab:~/CHF/Fabric-network$ peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: basic_1.0:d3ced865d65bb0db0980f7c27e023d7c8c7be01ebca4b37642b144
7dd5873818, Label: basic_1.0
```

Set the **package ID** as environment variable. (**Note**: Copy the Package ID value obtained using the **peer lifecycle chaincode queryinstalled** command. The given Package ID may not be the same) .

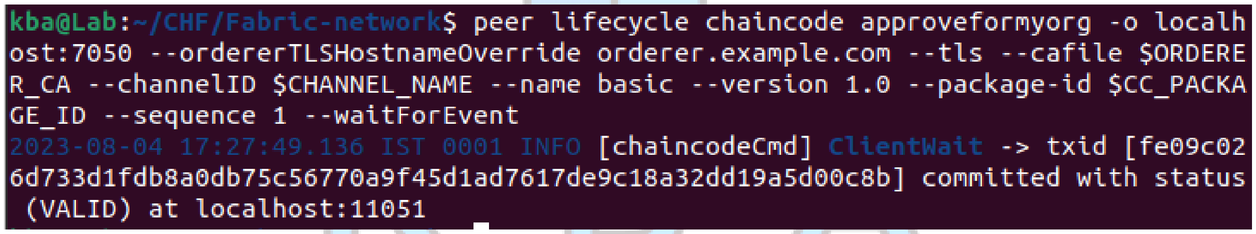
#####Execute the command in **peer0_Org3** terminal#####

export CC_PACKAGE_ID=basic_1.0:9919bbfa7aa2eade4bc8bc7bab939c3360220c115dbfc76027fcc4e3046e69d1

Then execute the **peer lifecycle chaincode approveformyorg** command.

#####Execute the command in **peer0_Org3** terminal#####

peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --waitForEvent

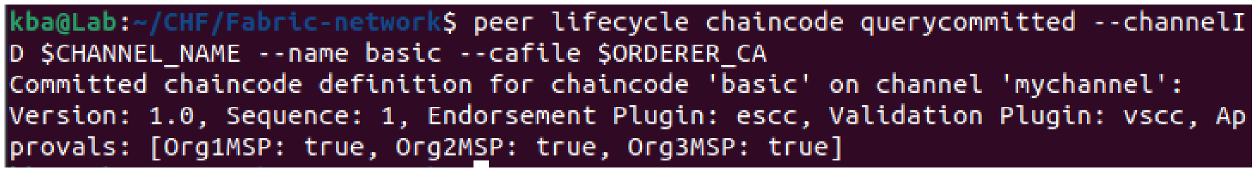


```
kba@Lab:~/CHF/Fabric-network$ peer lifecycle chaincode approveformyorg -o localh
ost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERE
R_CA --channelID $CHANNEL_NAME --name basic --version 1.0 --package-id $CC_PACKA
GE_ID --sequence 1 --waitForEvent
2023-08-04 17:27:49.136 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [fe09c02
6d733d1fdb8a0db75c56770a9f45d1ad7617de9c18a32dd19a5d00c8b] committed with status
 (VALID) at localhost:11051
```

Use the **peer lifecycle chaincode querycommitted** command to check if the chaincode definition you have approved has already been committed to the channel.

#####Execute the command in **peer0_Org3** terminal#####

peer lifecycle chaincode querycommitted --channelID $CHANNEL_NAME --name basic --cafile $ORDERER_CA

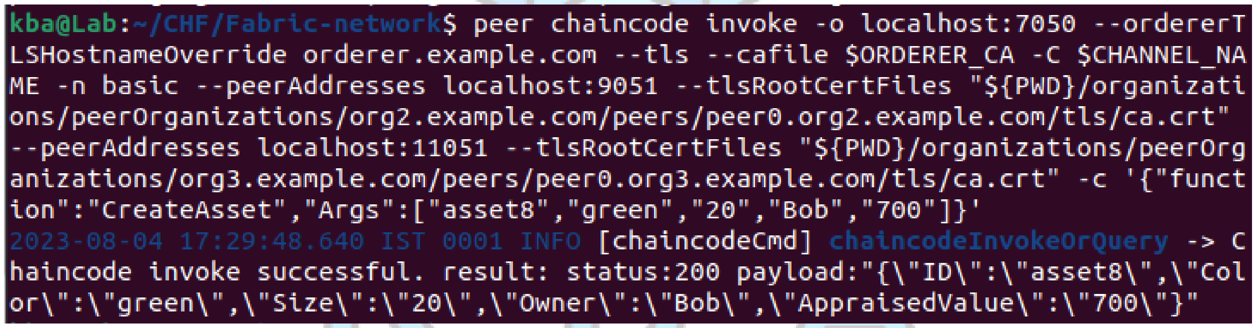


```
kba@Lab:~/CHF/Fabric-network$ peer lifecycle chaincode querycommitted --channelID $CHANNEL_NAME --name basic --cafile $ORDERER_CA
Committed chaincode definition for chaincode 'basic' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true]
```

Invoke the chaincode.

#####Execute the command in **peer0_Org3** terminal#####

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n basic --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt" -c '{"function":"CreateAsset","Args":["asset8","green","20","Bob","700"]}'

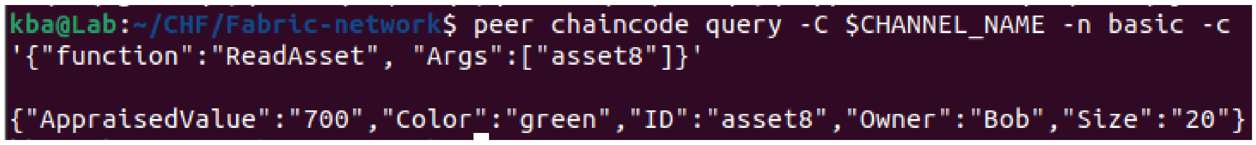


```
kba@Lab:~/CHF/Fabric-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n basic --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" --peerAddresses localhost:11051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt" -c '{"function":"CreateAsset","Args":["asset8","green","20","Bob","700"]}'
2023-08-04 17:29:48.640 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload:"{\"ID\":\"asset8\",\"Color\":\"green\",\"Size\":\"20\",\"Owner\":\"Bob\",\"AppraisedValue\":\"700\"}"
```

For querying we will be using the **peer chaincode query** command.

#####Execute the command in **peer0_Org3** terminal#####

peer chaincode query -C $CHANNEL_NAME -n basic -c '{"function":"ReadAsset", "Args":["asset8"]}'


```
kba@Lab:~/CHF/Fabric-network$ peer chaincode query -C $CHANNEL_NAME -n basic -c
'{"function":"ReadAsset", "Args":["asset8"]}'

{"AppraisedValue":"700","Color":"green","ID":"asset8","Owner":"Bob","Size":"20"}
```

Now we have successfully executed the chaincode lifecycle, submitted transaction and queried it from Org3.

Bring down the network

Execute the following commands to stop the network.

Note: Execute the following commands within the **Fabric-network** folder.

> #####Execute the command in **host** terminal#####
>
> **docker-compose -f docker/docker-compose-2org.yaml down**
>
> **docker-compose -f docker/docker-compose-ca.yaml down**
>
> **docker-compose -f docker/docker-compose-org3.yaml down**
>
> **docker volume rm $(docker volume ls -q)**

Remove all the crypto material generated for the network.

> #####Execute the command in **host** terminal#####
>
> **sudo rm -rf channel-artifacts/**
>
> **sudo rm -rf organizations/**
>
> **sudo rm basic.tar.gz**

Now, when you try the **docker ps** command, if any containers are listed, then execute the following commands.

> **docker rm $(docker container ls -q) --force**
>
> **docker system prune**

```
docker volume prune
```

```
docker network prune
```