1. Generate crypto material
    a. X.509 certs and signing keys are generated for all the entities
    b. Used for authenticating communication between these entities
    c. Uses **crypto-config.yaml**
    d. the generated certificates and keys will be saved to a folder titled crypto-config
    e. ../bin/cryptogen generate --config=./crypto-config.yaml
    f. The certs and keys (i.e. the MSP material) will be output into a directory - **crypto-config**

2. Configuration Transaction Generator
    a. **Configtxgen** tool is used, it creates four configuration artifacts
        i. orderer genesis block,
        ii. channel configuration transaction
        iii. two anchor peer transactions - one for each Peer Org
    b. Configtxgen consumes a file - **configtx.yaml**, hat contains the definitions for the sample network
        i. In **configtx.yaml** we specify the anchor peers for each Peer Org (peer0.org1.example.com & peer0.org2.example.com)
        ii. It also points to the location of the MSP directory for each member, in turn allowing us to store the root certificates for each Org in the orderer genesis block

    c. Generate Ordering service artifacts

```
../bin/configtxgen -profile TwoOrgsOrdererGenesis -channelID byfn-sys-channel
-outputBlock ./channel-artifacts/genesis.block
```

    d. Create channel configuration transaction

```
export CHANNEL_NAME=mychannel  && ../bin/configtxgen -profile TwoOrgsChannel
-outputCreateChannelTx ./channel-artifacts/channel.tx -channelID mychannel
```

    e. Define Anchor peer for org1

```
../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org1MSPanchors.tx -channelID mychannel -asOrg Org1MSP
```

    f. Define Anchor peer for org2

```
../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate
./channel-artifacts/Org2MSPanchors.tx -channelID mychannel -asOrg Org2MSP
```

    g. Artifacts generated above are stored in **channel-artifacts** folder

3. Start the network

```
docker-compose -f docker-compose-cli.yaml up -d
```

4. Create and join channel
    a. Enter cli docker

```
docker exec -it cli bash
```

b. Pass channel configuration to orderer

```
export CHANNEL_NAME=mychannel

peer channel create -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/channel.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

c. Above command creates mychannel.block file, that will be used to join the channel

```
peer channel join -b mychannel.block
```

d. Set env variables for anchor peer of second organisation and join that to the channel

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

peer channel join -b mychannel.block
```

5. Update anchor peers

   a. First for org2, since env variables in last command are updated for org2

```
peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

   b. Now set env variables for org1

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:9051
CORE_PEER_LOCALMSPID="Org1MSP"
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
```

c. Update channel definition for org1

```
peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

6. Install and Instantiate chaincode

```
peer chaincode install -n mycc -v 1.0 -l node -p /opt/gopath/src/github.com/chaincode/chaincode_example02/node/
```

a. Modify env variables for org2

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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
```

b. Install chaincode on peer2

```
peer chaincode install -n mycc -v 1.0 -l node -p /opt/gopath/src/github.com/chaincode/chaincode_example02/node/
```

c. Instantiate chaincode, command will
   i. Instantiate chaincode on the channel
   ii. Set endorsement policy on chaincode
   iii. Launch chaincode container on the peers

```
peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -l node -v 1.0 -c '{"Args":["init","a", "100", "b","200"]}' -P "AND ('Org1MSP.peer','Org2MSP.peer')"
```

d. Query chanicode

```
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
```

    e. Invoke chaincode

```
peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["invoke","a","b","10"]}'
```

7. Some observations
    a. To perform read write operations, chaincode must be installed on a peer
    b. Chaincode container is started only after it is initialised (First transaction is called)
    c. To check logs for chaincode installation and instantiation
        i. docker logs -f cli
    d. To check chaincode logs
        i. docker logs <chaincode container name / id>
        ii. docker logs <peer name / id>

8. To use **couchDB** instead of default option **levelDB**
    a. docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -d
    b. CouchDB is noSQL database
    c. Can perform rich queries from chaincode
9. To bring down the network
    a. ./byfn.sh down