```
#!/bin/bash

echo
echo "  ____    _____      _      ____    _____ "
echo "/ ___|  |_   _|    / \    |  _ \  |_   _|"
echo "\___ \    | |     / _ \   | |_) |   | |  "
echo " ___) |   | |    / ___ \  |  _ <    | |  "
echo "|____/    |_|   /_/   \_\ |_| \_\   |_|  "
echo
echo "//// Build 3 channels network with 4 peers ////"
echo
CHANNEL_NAME_BASE="$1"
DELAY="$2"
: ${CHANNEL_NAME_BASE:="channel"}
: ${DELAY:=1}
: ${TIMEOUT:="2"}
COUNTER=1
MAX_RETRY=5
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganiza
tions/zak.codes/orderers/orderer.zak.codes/msp/tlscacerts/tlsca.zak.codes-cert.pem

echo "Channel name : "$CHANNEL_NAME_BASE"1"
echo "Channel name : "$CHANNEL_NAME_BASE"2"
echo "Channel name : "$CHANNEL_NAME_BASE"3"

# verify the result of the end-to-end test
verifyResult () {
  if [ $1 -ne 0 ] ; then
    echo "!!!!!!!!!!!!!!! "$2" !!!!!!!!!!!!!!!!"
    echo "========= ERROR !!! FAILED to execute End-2-End Scenario ==========="
    echo
      exit 1
  fi
}

setGlobals () {

  if [ $1 -eq 0 -o $1 -eq 1 ] ; then
    CORE_PEER_LOCALMSPID="City1MSP"
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/city1.zak.codes/peers/peer0.city1.zak.codes/tls/ca.crt
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypt
o/peerOrganizations/city1.zak.codes/users/Admin@city1.zak.codes/msp
    if [ $1 -eq 0 ]; then
      CORE_PEER_ADDRESS=peer0.city1.zak.codes:7051
    else
      CORE_PEER_ADDRESS=peer1.city1.zak.codes:7051
      CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/cry
pto/peerOrganizations/city1.zak.codes/users/Admin@city1.zak.codes/msp
    fi
  else
    CORE_PEER_LOCALMSPID="City2MSP"
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/c
rypto/peerOrganizations/city2.zak.codes/peers/peer0.city2.zak.codes/tls/ca.crt
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypt
o/peerOrganizations/city2.zak.codes/users/Admin@city2.zak.codes/msp
    if [ $1 -eq 2 ]; then
      CORE_PEER_ADDRESS=peer0.city2.zak.codes:7051
    else
      CORE_PEER_ADDRESS=peer1.city2.zak.codes:7051
    fi
  fi

  env |grep CORE
}

createChannel() {
  setGlobals 0

    if [ -z "$CORE_PEER_TLS_ENABLED" -o "$CORE_PEER_TLS_ENABLED" = "false" ]; then
    peer channel create -o orderer.zak.codes:7050 -c $CHANNEL_NAME -f ./channel-arti
facts/${CHANNEL_NAME}.tx >&log.txt
  else
```

```
    peer channel create -o orderer.zak.codes:7050 -c $CHANNEL_NAME -f ./channel-arti
facts/${CHANNEL_NAME}.tx --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA >&log.txt
  fi
  res=$?
  cat log.txt
  verifyResult $res "Channel creation failed"
  echo "===================== Channel \"$CHANNEL_NAME\" is created successfully ====
================= "
  echo
}

updateAnchorPeers() {
  PEER=$1
  setGlobals $PEER

  if [ -z "$CORE_PEER_TLS_ENABLED" -o "$CORE_PEER_TLS_ENABLED" = "false" ]; then
    peer channel update -o orderer.zak.codes:7050 -c $CHANNEL_NAME -f ./channel-arti
facts/${CHANNEL_NAME}${CORE_PEER_LOCALMSPID}anchors.tx >&log.txt
  else
    peer channel update -o orderer.zak.codes:7050 -c $CHANNEL_NAME -f ./channel-arti
facts/${CHANNEL_NAME}${CORE_PEER_LOCALMSPID}anchors.tx --tls $CORE_PEER_TLS_ENABLED
--cafile $ORDERER_CA >&log.txt
  fi
  res=$?
  cat log.txt
  verifyResult $res "Anchor peer update failed"
  echo "===================== Anchor peers for org \"$CORE_PEER_LOCALMSPID\" on \"$C
HANNEL_NAME\" is updated successfully ===================== "
  sleep $DELAY
  echo
}

## Sometimes Join takes time hence RETRY atleast for 5 times
joinWithRetry () {
  peer channel join -b $CHANNEL_NAME.block  >&log.txt
  res=$?
  cat log.txt
  if [ $res -ne 0 -a $COUNTER -lt $MAX_RETRY ]; then
    COUNTER=` expr $COUNTER + 1`
    echo "PEER$1 failed to join the channel, Retry after 2 seconds"
    sleep $DELAY
    joinWithRetry $1
  else
    COUNTER=1
  fi
  verifyResult $res "After $MAX_RETRY attempts, PEER$ch has failed to Join the Chann
el"
}

joinChannel () {
  for peer in 0 1 2 3; do
    setGlobals $peer
    joinWithRetry $peer
    echo "===================== PEER$peer joined on the channel \"$CHANNEL_NAME\" ==
=================== "
    sleep $DELAY
    echo
  done
}

installChaincode () {
  PEER=$1
  CC_NAME=$2
  PATH_TO_CC=$3
  setGlobals $PEER
  peer chaincode install -n $CC_NAME -v 1.0 -p $PATH_TO_CC >&log.txt
  res=$?
  cat log.txt
        verifyResult $res "Chaincode installation on remote peer PEER$PEER has Faile
d"
  echo "===================== Chaincode $CC_NAME is installed on remote peer PEER$PE
ER ===================== "
  echo
```

```
}

instantiateChaincode () {
  PEER=$1
  CC_NAME=$2
  # PP=$3 # policy and payload code for channel number
  setGlobals $PEER
  echo $PP
  # while 'peer chaincode' command can get the orderer endpoint from the peer (if jo
in was successful),
  # lets supply it directly as we know it using the "-o" option

  if [ -z "$CORE_PEER_TLS_ENABLED" -o "$CORE_PEER_TLS_ENABLED" = "false" ]; then
    peer chaincode instantiate -o orderer.zak.codes:7050 -C $CHANNEL_NAME -n $CC_NAM
E -v 1.0 -c "${PAYLOAD}" -P "${POLICY}" >&log.txt
  else
    peer chaincode instantiate -o orderer.zak.codes:7050 --tls $CORE_PEER_TLS_ENABLE
D --cafile $ORDERER_CA -C $CHANNEL_NAME -n $CC_NAME -v 1.0 -c "${PAYLOAD}" -P "${POL
ICY}" >&log.txt
  fi
  res=$?
  cat log.txt
  verifyResult $res "Chaincode instantiation on PEER$PEER on channel '$CHANNEL_NAME'
 failed"
 echo "===================== Chaincode Instantiation on PEER$PEER on channel '$CHAN
NEL_NAME' is successful ===================== "
  echo
}

chaincodeInvoke () {
  PEER=$1
  CC_NAME=$2
  setGlobals $PEER
  # while 'peer chaincode' command can get the orderer endpoint from the peer (if jo
in was successful),
  # lets supply it directly as we know it using the "-o" option
  if [ -z "$CORE_PEER_TLS_ENABLED" -o "$CORE_PEER_TLS_ENABLED" = "false" ]; then
    peer chaincode invoke -o orderer.zak.codes:7050 -C $CHANNEL_NAME -n $CC_NAME -c
"${PAYLOAD}" >&log.txt
  else
    peer chaincode invoke -o orderer.zak.codes:7050  --tls $CORE_PEER_TLS_ENABLED --
cafile $ORDERER_CA -C $CHANNEL_NAME -n $CC_NAME -c "${PAYLOAD}" >&log.txt
  fi
  res=$?
  cat log.txt
  verifyResult $res "Invoke execution on PEER$PEER failed "
  # Extract the returned payload without quotes
  RETURNED_PAYLOAD=$(cat log.txt | awk -F"payload:" '{print $2}')
    RETURNED_PAYLOAD=$(echo $RETURNED_PAYLOAD | awk -F">" '{print $1}')
  echo "===================== Invoke transaction on PEER$PEER on channel '$CHANNEL_N
AME' and chaincode '$CC_NAME' is successful ===================== "
  echo
}

## Create channels
echo "Creating channels..."
for i in 1 2 3; do
  CHANNEL_NAME="${CHANNEL_NAME_BASE}${i}"
  createChannel
  echo "===================== the channel \"$CHANNEL_NAME\" is created ============
======== "
  echo
done

## Join all the peers to the channel
echo "Having all peers join the channel..."
CHANNEL_NAME="${CHANNEL_NAME_BASE}1"
for i in 0 1; do
  setGlobals $i
  joinWithRetry $i
  echo "===================== PEER$i joined on the channel \"$CHANNEL_NAME\" =======
============== "
  sleep $DELAY
  echo
done

for i in 2 3; do
  CHANNEL_NAME="${CHANNEL_NAME_BASE}${i}"
  joinChannel
done

## Set the anchor peers for each org in the channel
CHANNEL_NAME="${CHANNEL_NAME_BASE}1"
echo "Updating anchor peers for City1 for channel '${CHANNEL_NAME}' ..."
updateAnchorPeers 0
for i in 2 3; do
  CHANNEL_NAME="${CHANNEL_NAME_BASE}${i}"
  echo "Updating anchor peers for City1 for channel '${CHANNEL_NAME}' ..."
  updateAnchorPeers 0
  echo "Updating anchor peers for City2 for channel '${CHANNEL_NAME}' ..."
  updateAnchorPeers 2
done

## Install chaincode_data on Peer0/City1 and Peer2/City2
echo "--> Installing chaincode_data on Peer0/City1..."
echo
installChaincode 0 chaincode_data github.com/hyperledger/fabric/chaincode/chaincode_
data

# Install chaincode_ad on Peer0/City1...
echo "--> Installing chaincode_ad on Peer0/City1..."
echo
installChaincode 0 chaincode_ad github.com/hyperledger/fabric/chaincode/chaincode_ad

# Install chaincode_ad on Peer2/City2...
echo "--> Install chaincode_ad on Peer2/City2..."
echo
installChaincode 2 chaincode_ad github.com/hyperledger/fabric/chaincode/chaincode_ad

# Install chaincode_tokens on Peer0/City1...
echo "--> Installing chaincode_tokens on Peer0/City1..."
echo
installChaincode 0 chaincode_tokens github.com/hyperledger/fabric/chaincode/chaincod
e_tokens

# Install chaincode_tokens on Peer2/City2...
echo "--> Installing chaincode_tokens on Peer2/City2..."
echo
installChaincode 2 chaincode_tokens github.com/hyperledger/fabric/chaincode/chaincod
e_tokens

#Instantiate chaincode_data on Peer0/City1
echo "--> Instantiating chaincode_data on Peer0/City1..."
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}1"
PAYLOAD='{"Args":["1"]}'
POLICY="AND ('City1MSP.member')"
instantiateChaincode 0 chaincode_data

#Instantiate chaincode_ad on Peer2/City2
echo "--> Instantiating chaincode_ad on Peer0/City1..."
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}2"
PAYLOAD='{"Args":["1"]}'
POLICY="OR ('City1MSP.member','City2MSP.member')"
instantiateChaincode 0 chaincode_ad

#Instantiate chaincode_tokens on Peer0/City1
echo "--> Instantiating chaincode_tokens on Peer0/City1..."
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["1000000"]}'
POLICY="OR ('City1MSP.member','City2MSP.member')"
# switch to peer 0 on City 1 so it owns account 1
instantiateChaincode 0 chaincode_tokens
```

```bash
# Create global variable RETURNED_PAYLOAD that is used in chaincodeInvoke function
RETURNED_PAYLOAD=""

# Invoke on chaincode_data on Peer0/City1
echo "--> Sending invoke first transaction createData on City1/peer0 on chaincode_da
ta ..."
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}1"
PAYLOAD='{"Args":["createData", "1", "test data", "50", "Celsius", "20180321163750",
 "marcel"]}'
chaincodeInvoke 0 chaincode_data

# Invoke on chaincode_ad on Peer0/City1
# Free data
echo "--> Sending invoke first transaction createDataEntryAd on Peer0/City1 on chain
code_ad ..."
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}2"
PAYLOAD='{"Args":["createDataEntryAd", "1", "test data", "50", "Celsius", "201803211
63750", "marcel", "0", "2"]}'
chaincodeInvoke 0 chaincode_ad

# Invoke on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction createAccount on Peer2/City2 on chaincode_token
s"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["createAccount", "2", "Test_Account_City2"]}'
# switch to peer 2 on City2 so it owns the account
sleep 3
chaincodeInvoke 2 chaincode_tokens

# Invoke on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction sendTokensSafe on Peer0/City1 on chaincode_toke
ns"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["sendTokensSafe", "1", "2", "50000", "false"]}'
# switch to peer 0 on City1 because that is the owner of the account
sleep 3
chaincodeInvoke 0 chaincode_tokens

# Invoke on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction updateAccountTokens on Peer0/City1 on chaincode
_tokens"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["updateAccountTokens", "1"]}'
chaincodeInvoke 0 chaincode_tokens

# Invoke on chaincode_tokens on Peer0/City1
echo "--> Sending invoke transaction updateAccountTokens on Peer0/City1 on chaincode
_tokens"
echo
CHANNEL_NAME="${CHANNEL_NAME_BASE}3"
PAYLOAD='{"Args":["updateAccountTokens", "2"]}'
chaincodeInvoke 0 chaincode_tokens

# peer chaincode invoke --tls true --cafile /opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/ordererOrganizations/zak.codes/orderers/orderer.zak.codes/msp/tlsc
acerts/tlsca.zak.codes-cert.pem -n chaincode_ad -c '{"Args":["revealPaidData", "chan
nel1", "chaincode_data", "2", "20180321160000", "channel3", "chaincode_tokens", "txI
D"]}' -C channel2

echo
echo " _____   _   _   ____   "
echo "| ____| | \ | | |  _ \  "
echo "|  _|    |  \| | | | | |  "
echo "| |___  | |\  | | |_| |  "
echo "|_____| |_| \_| |____/  "
echo
echo "//// Build 3 channels network with 4 peers successfully ////"
echo
```

```
exit 0
```