Fabric Network

test-network Architecture

- No of Orgs : 2
  - Org1 - 1 peer (peer0.org1.example.com)
  - Org2 - 1 peer (peer0.org2.example.com)
- Ordering Service : Raft (Single Node - orderer.example.com)
- Database Type - CouchDB (couchdb0, couchdb1)
- Certificate Authority : Separate CA for org1, org2 and orderer
  - ca_org1
  - ca_org2
  - ca_orderer

Steps to set-up a fabric network

- Build the network
  - Generate the crypto material (fabric-ca or cryptogen)
  - Bring up the components (2 orgs with 1 peer each and 1 orderer)
  - Generate the genesis block file
  - Create channel and join orderer
  - Join the peers to the channel
  - Anchor peer update
- Deploy the chaincode
  - Package the chaincode
  - Install the packaged chaincode to selected peers
  - Approve chaincode with chaincode definition
  - Commit the chaincode to the channel
- Invoke/Query the chaincode

Dive onto the files

There are five important files to bootstrap the network.

- **docker-compose-ca.yaml** - Used to generate the certificates for the organizations.
- **registerEnroll.sh** - Script file used to register and enroll users, and organize the certificates.
- **configtx.yaml** - Defines the initial network configuration.
- **docker-compose-2org.yaml** - Used to define the containers of Network.
- **core.yaml** - Defines the peer configurations and used for executing peer commands

docker-compose-2org.yml

The docker-compose file will be having 6 services

- 2 organization peers (1 peer for each org)
- 2 couch db (1 for each peer)
- Orderer

Container definitions handle:

- Port configuration
- Establishment of dependencies
- Environment variables (in regard to the specified component)
- Path to crypto materials
- Volumes and mapping.

Environment variables - Peer

CORE_PEER_ID
CORE_PEER_ADDRESS
CORE_PEER_MSPCONFIGPATH
CORE_PEER_LOCALMSPID
CORE_PEER_TLS_ENABLED
CORE_PEER_TLS_ROOTCERT_FILE
CORE_PEER_TLS_CERT_FILE
CORE_PEER_TLS_KEY_FILE

Environment variables - Couch DB

COUCHDB_USER
COUCHDB_PASSWORD

Peer:
CORE_LEDGER_STATE_STATEDATABASE
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS
CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME
CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD

Environment variables - Orderer

ORDERER_GENERAL_LOCALMSPID
ORDERER_GENERAL_LOCALMSPDIR
ORDERER_GENERAL_GENESISFILE
ORDERER_GENERAL_TLS_ENABLED
ORDERER_GENERAL_TLS_PRIVATEKEY
ORDERER_GENERAL_TLS_CERTIFICATE
ORDERER_GENERAL_TLS_ROOTCAS

configtx.yaml

Define configtx.yaml file, which will have all the network related configurations

- Organizations
  - Name (informal name used to identify the organization)
  - MSP ID and path (MSP ID acts as a unique identifier for the organization)
  - Policies
- Capabilities
  - Channel
  - Orderer
  - Application
- Application
- Orderer
- Channel
- Profiles

Generating the channel artifacts

- Generate the genesis block for channel

  configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/${CHANNEL_NAME}.block -channelID $CHANNEL_NAME

- Create the application channel and join the orderer

  osnadmin channel join --channelID $CHANNEL_NAME --config-block ./channel-artifacts/$CHANNEL_NAME.block -o localhost:7053 --ca-file $ORDERER_CA --client-cert $ORDERER_ADMIN_TLS_SIGN_CERT --client-key $ORDERER_ADMIN_TLS_PRIVATE_KEY

Join channel

- Join the peer to a channel
  Command: **peer channel join**
  Eg: peer channel join -b ./channel-artifacts/$CHANNEL_NAME.block

- List of channels the peer has joined
  Command: **peer channel list**

Anchor Peer Update

- peer channel fetch config channel-artifacts/config_block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL_NAME --tls --cafile $ORDERER_CA
- 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
- cp config.json config_copy.json
- jq '.channel_group.groups.Application.groups.Org1MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "peer0.org1.example.com","port": 7051}]},"version": "0"}}' config_copy.json > modified_config.json

Anchor Peer Update

- configtxlator proto_encode --input config.json --type common.Config --output config.pb
- configtxlator proto_encode --input modified_config.json --type common.Config --output modified_config.pb
- configtxlator compute_update --channel_id ${CHANNEL_NAME} --original config.pb --updated modified_config.pb --output config_update.pb
- configtxlator proto_decode --input config_update.pb --type common.ConfigUpdate --output config_update.json
- echo '{"payload":{"header":{"channel_header":{"channel_id":"$CHANNEL_NAME", "type":2}},"data":{"config_update":'$(cat config_update.json)'}}}' | jq . > config_update_in_envelope.json
- configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope --output config_update_in_envelope.pb
- peer channel update -f channel-artifacts/config_update_in_envelope.pb -c $CHANNEL_NAME -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA

Deploy Chaincode

- Package the Chaincode
  Command: **peer lifecycle chaincode package**
  Eg: peer lifecycle chaincode package basic.tar.gz --path ../Chaincode/chaincode-go/ --lang golang --label basic_1.0

- Install the Chaincode
  Command: **peer lifecycle chaincode install**
  Eg: peer lifecycle chaincode install basic.tar.gz
  Returns a package ID

- Query the installed chaincodes on a peer
  Command: **peer lifecycle chaincode queryinstalled**

Approve chaincode

- Chaincode needs to be approved by the organization peer
  Command: **peer lifecycle chaincode approveformyorg**
  Eg: : peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID $CHANNEL_NAME --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile $ORDERER_CA --waitForEvent

- To define **new endorsement policy** use the flag **--signature-policy**
  Eg: --signature-policy "OR('Org1MSP.peer')"

- To implement PDC use the flag **--collections-config**
  Eg: --collections-config ../Chaincode/KBA-Automobile/collection.json

Commit chaincode

- Commit the chaincode definition on the channel

  Command: **peer lifecycle chaincode commit**

  Eg: peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID $CHANNEL_NAME --name basic --version 1.0 --sequence 1 --tls --cafile $ORDERER_CA --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt"

- If needed include signature-policy, collections-config

Invoke Chaincode

- Command: **peer chaincode invoke**
- Eg: peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $ORDERER_CA -C $CHANNEL_NAME -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'

Query chaincode

- Command: **peer chaincode query**

  Eg: peer chaincode query -C $CHANNEL_NAME -n basic -c '{"Args":["GetAllAssets"]}'

THANK YOU