Fabric Network

test-network Architecture

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

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
  - Joining the peers to the created 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 for building the network.

- **docker-compose-ca.yaml** - Starts the fabric-ca server.
- **registerEnroll.sh** - Script file used to register and enroll users, and organize the certificates.
- **docker-compose-2org.yaml** - Used to define the containers of Network components.
- **configtx.yaml** - Defines the initial network configuration.
- **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

- Orderer
- 2 Peers (1 peer for each org)
- 2 couch db (1 for each peer)
- cli

Container definitions include:

- Image
- Environment variables
  - Path to crypto materials
  - Listening ports
- Port mapping
- Volume mounting
- Establishment of dependencies

Environment variables - Orderer

FABRIC_LOGGING_SPEC
ORDERER_GENERAL_LISTENPORT
ORDERER_GENERAL_LISTENADDRESS
ORDERER_GENERAL_LOCALMSPID
ORDERER_GENERAL_LOCALMSPDIR
ORDERER_GENERAL_TLS_ENABLED
ORDERER_GENERAL_TLS_PRIVATEKEY
ORDERER_GENERAL_TLS_CERTIFICATE
ORDERER_GENERAL_TLS_ROOTCAS

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

configtx.yaml

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

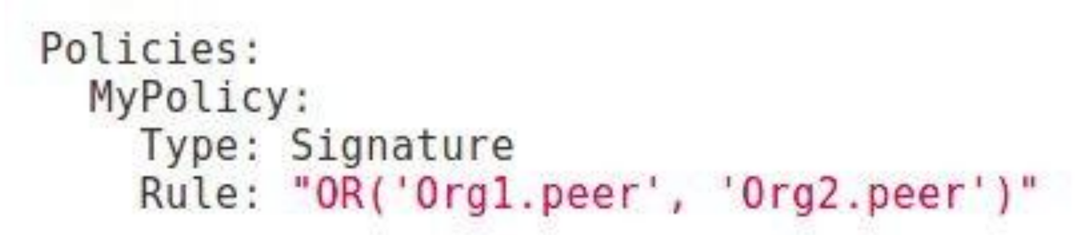
- Organizations
  - Name (informal name used to identify the organization)
  - MSP ID and path (MSP ID acts as a unique identifier for the organization)
  - Policies
- Capabilities
  - Channel
  - Orderer
  - Application
- Application
- Orderer
- Channel
- Profiles

Policies: Two Types

- Signature Policy
- Implicit Meta

Policies: Signature Policies

➔ These policy identify specific users who must sign in order for a policy to be satisfied.



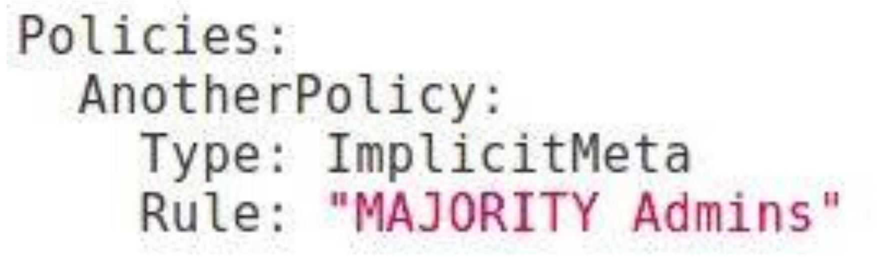
```
Policies:
  MyPolicy:
    Type: Signature
    Rule: "OR('Org1.peer', 'Org2.peer')"
```

➔ The above policy can be interpreted as the policy name “MyPolicy” can only be satisfied by the signature of an identity with role of “ a peer from Org1” or “ a peer from Org2”

Signature policies support Arbitrary combinations of AND, OR and NOutOf

ImplicitMeta

➔ ImplicitMeta policies aggregate the result of policies deeper in the configuration hierarchy that are ultimately defined by Signature Policies.



```
Policies:
  AnotherPolicy:
    Type: ImplicitMeta
    Rule: "MAJORITY Admins"
```

➔ Here, the policy "AnotherPolicy" can be satisfied by the Majority of Admins, where Admins is eventually being specified by the lower level " Signature-policy"

The expressions used are "ALL", "ANY", "MAJORITY"

Generating the channel artifacts

- Generate the genesis block for channel

```
configtxgen -profile ChannelUsingRaft -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

- **core.yaml** file needed for executing peer commands

- 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

Anchor Peer Update

- 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

THANK YOU

K B A