**Hyperledger fabric Guide**

**By.LJK**

1. **운용 환경 설정**
   1. 가이드 작성 환경
      1. OS : Centos Linux 7.8.2003
      2. Hardware : Naver cloud [Compact]
   2. Docker 설치
      1. 2020-06-18 latest version : docker engine 19.03.12
   3. Docker-compose 설치
      1. 2020-07-02 latest version : 1.26.2
   4. Go lang 설치
      1. 2020-08-11 latest version : 1.1
   5. (Optional)필요한 경우 node.js 및 java 설치
   6. 프로젝트를 위한 별도의 프로젝트 폴더 생성
2. **바이너리 파일 및 도커 이미지 다운로드**
   1. Fabric-samples 다운
      1. Curl 명령어 : curl -sSL https://bit.ly/2ysbOFE | bash -s

(hyperledger fabric latest version 다운로드)

\* old version 다운로드 : curl -sSL https://bit.ly/2ysbOFE | bash -s -- <fabric\_version> <fabric-ca\_version>

* 1. 도커 이미지 확인
     1. docker images 명령어 수행
     2. 설치된 이미지 확인
  2. 바이너리 파일 복사
     1. Fabric-samples 폴더의 bin 폴더 전부를 작성 중인 프로젝트 폴더로 복사
     2. Hyperledger fabric 2.2 version 기준 bin 폴더에 9개의 파일 확인
        1. configtxgen \*
        2. configtxlator
        3. cryptogen \*
        4. discover
        5. fabric-ca-client \*
        6. fabric-ca-server \*
        7. idemixgen
        8. orderer
        9. peer

(\* 파일이 존재하면 진행 가능)

※ 여기까지 진행이 완료되면 hyperledger fabric 구동을 위한 초기 설정 및 구성 완료

1. **hyperledger fabric MSP 생성**
   1. crypto-config.yaml 작성
      1. orderer 조직과 부분과 peer 조직을 작성
      2. 해당 설정 파일로 hyperledger fabric 초기 참가 노드를 설정
   2. bin의 cryptogen을 이용해 crypto-config 생성
   3. crypto-config 폴더 내 MSP 생성물 확인

|  |
| --- |
| OrdererOrgs:    - Name: Orderer      Domain: example.com      EnableNodeOUs: true      Specs:        - Hostname: orderer1          SANS:            - localhost        - Hostname: orderer2          SANS:            - localhost        - Hostname: orderer3          SANS:            - localhost  PeerOrgs:      - Name: Org1        Domain: org1.example.com        EnableNodeOUs: true        Template:          Count: 1          SANS:            - localhost        Users:          Count: 1 |

* 1. crypto-conig.yaml 템플릿
  2. 실행 명령어
     1. ./cryptogen generate --config=[crypto-config.yaml 파일] –-output=[ 원하는 crypto-config 폴더 위치]

1. **네트워크 구동을 위한 설정 파일 생성**
   1. Configyx.yaml 작성
      1. 해당 설정 파일로 genesis block 및 channel tx을 정의
      2. 만들어진 결과물로 오더러 부팅과 실제 채널을 만들게 됨
   2. bin의 configtxgen으로 genesis.block 생성
      1. hyperledger fabric의 orderer가 해당 genesis block으로 부팅
   3. bin의 configtxgen으로 channel.tx 생성
      1. 해당 tx으로 peer가 실제 channel과 channel block을 생성
   4. bin의 configtxgen으로 anchor.tx 생성[현재 configtxlator로 인해 해당 기능 미사용 권장]
      1. 원래 해당 기능으로 만들어진 anchor.tx로 실제 생성된 채널을 업데이트 함
      2. 다만 현재 configtxlator라는 실행 파일로 해당 기능을 대체했기 때문에 사용을 권장하지는 않지만 해당 실행 파일에 대해서는 구체적인 이용법 조사 필요

|  |
| --- |
| Organizations:      - &OrdererOrg          Name: OrdererOrg          ID: OrdererMSP          MSPDir: ../crypto-config/ordererOrganizations/example.com/msp          Policies:              Readers:                  Type: Signature                  Rule: "OR('OrdererMSP.member')"              Writers:                  Type: Signature                  Rule: "OR('OrdererMSP.member')"              Admins:                  Type: Signature                  Rule: "OR('OrdererMSP.admin')"          OrdererEndpoints:              - orderer1.example.com:7050              - orderer2.example.com:8050              - orderer3.example.com:9050      - &Org1          Name: Org1Org          ID: Org1MSP          MSPDir: ../crypto-config/peerOrganizations/org1.example.com/msp          Policies:              Readers:                  Type: Signature                  Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"              Writers:                  Type: Signature                  Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"              Admins:                  Type: Signature                  Rule: "OR('Org1MSP.admin')"              Endorsement:                  Type: Signature                  Rule: "OR('Org1MSP.peer')"          AnchorPeers:              - Host: peer0.org1.example.com                Port: 7051  Capabilities:      Channel: &ChannelCapabilities          V2\_0: true      Orderer: &OrdererCapabilities          V2\_0: true      Application: &ApplicationCapabilities          V2\_0: true  Application: &ApplicationDefaults      Organizations:      Policies:          Readers:              Type: ImplicitMeta              Rule: "ANY Readers"          Writers:              Type: ImplicitMeta              Rule: "ANY Writers"          Admins:              Type: ImplicitMeta              Rule: "MAJORITY Admins"          LifecycleEndorsement:              Type: ImplicitMeta              Rule: "MAJORITY Endorsement"          Endorsement:              Type: ImplicitMeta              Rule: "MAJORITY Endorsement"      Capabilities:          <<: \*ApplicationCapabilities  Orderer: &OrdererDefaults      OrdererType: etcdraft      EtcdRaft:          Consenters:          - Host: orderer1.example.com            Port: 7050            ClientTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer1.example.com/tls/server.crt            ServerTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer1.example.com/tls/server.crt          - Host: orderer2.example.com            Port: 8050            ClientTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/tls/server.crt            ServerTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/tls/server.crt          - Host: orderer3.example.com            Port: 9050            ClientTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer3.example.com/tls/server.crt            ServerTLSCert: ../crypto-config/ordererOrganizations/example.com/orderers/orderer3.example.com/tls/server.crt      BatchTimeout: 2s      BatchSize:          MaxMessageCount: 10          AbsoluteMaxBytes: 99 MB          PreferredMaxBytes: 512 KB      Organizations:      Policies:          Readers:              Type: ImplicitMeta              Rule: "ANY Readers"          Writers:              Type: ImplicitMeta              Rule: "ANY Writers"          Admins:              Type: ImplicitMeta              Rule: "MAJORITY Admins"          BlockValidation:              Type: ImplicitMeta              Rule: "ANY Writers"  Channel: &ChannelDefaults      Policies:          Readers:              Type: ImplicitMeta              Rule: "ANY Readers"          Writers:              Type: ImplicitMeta              Rule: "ANY Writers"          Admins:              Type: ImplicitMeta              Rule: "MAJORITY Admins"      Capabilities:          <<: \*ChannelCapabilities  Profiles:      OneOrgsOrdererGenesis:          <<: \*ChannelDefaults          Orderer:              <<: \*OrdererDefaults              Organizations:                  - \*OrdererOrg              Capabilities:                  <<: \*OrdererCapabilities          Consortiums:              SampleConsortium:                  Organizations:                      - \*Org1        OneOrgsChannel:          Consortium: SampleConsortium          <<: \*ChannelDefaults          Application:              <<: \*ApplicationDefaults              Organizations:                  - \*Org1              Capabilities:                  <<: \*ApplicationCapabilities |

* 1. Configtx.yaml 템플릿
  2. 실행 명령어
     1. ./configtxgen –configPath [configtx.yaml 파일 위치(폴더까지만)] –profile [configtx.yaml 내 genesis profile 부분] –channelID [오더러 채널 이름] –outputBlock [genesis.block이 생성될 원하는 위치]
     2. ./configtxgen –configPath [configtx.yaml 파일 위치(폴더까지만)] –profile [configtx,yaml 내 channel profile 부분] –channelID [채널 이름] –outputCreateChannelTx [channel.tx이 생성될 원하는 위치]

1. **Hyperledger fabric 네트워크 구동**
   1. docker-compose.yaml 작성
      1. docker container를 구동하기 위한 파일
      2. 한번에 다수의 컨테이너를 구동할 수 있는 기능이 docker-compose
   2. docker-compose.yaml은 크게 orderer 부분, peer 부분, ca 부분, couchdb 부분, cli 부분으로 나뉠 수 있으며 각 부분은 다른 docker-compose.yaml 파일로 작성 할 수 있음
   3. 멀티 호스트로 네트워크를 구성하기 위해서는 docker-swarm을 사용해야 하며 해당 기능과 관련한 설정도 docker-compose.yaml에 작성
   4. 작성된 docker-compose.yaml로 docker 실행
      1. Docker-compose –f [docker-compose.yaml 파일] up –d
      2. Docker가 실행되고나면 docker ps –a 명령어로 모든 컨테이너가 정상적으로 작동하는 지 확인
      3. 문제가 발생하면 docker logs [컨테이너 ID or 컨테이너 이름]로 에러 로그 추적
2. **Channel 생성**
   1. Hyperledger fabric cli 내부에 접속하거나 외부에서 docker exec 명령어로 cli 작동
      1. 내부 접속은 docker exec –it cli /bin/bash로 가능
      2. 외부 수행 또한 docker exec –it cli ~로 가능
   2. channel.tx를 이용해 실제 channel을 생성
      1. docker의 volume 설정을 이용해 host os의 channel.tx를 peer container 내부로 가져올 수 있음
      2. channel 생성 또한 하나의 거래로 볼 수 있기 때문에 orderer로 전송을 해야함
         1. orderer의 TLS 인증서가 필요함
         2. orderer는 docker 부팅 시 RAFT 라운드를 시작하고 있음-> orderer가 시작했다는 의미
      3. channel이 만들어지면 channel에 참가하기로 한 노드들을 참여시켜야 함
         1. channel 참가 노드들은 configtx.yaml에 정의되어 있음
         2. configtx.yaml을 이용해 만든 channel.tx에도 채널에 관한 설정이 기록되어 있음
      4. 채널 생성 명령어
         1. (cli 외부에서 실행) docker exec [cli 컨테이너 이름] peer channel create –c [channel.tx 설정 시 지정한 채널 이름] –o [오더러 도메인:포트번호] –f [컨테이너 내부 channel.tx 위치] –-tls –-cafile [오더러 MSP tls 인증서 위치]
         2. 위의 명령어를 통해 [채널이름].block이 만들어짐
         3. (cli 외부에서 실행) docker exec [cli 컨테이너 이름] peer channel join –b [체널이름.block 위치]
      5. 과정이 끝나면 현재 채널이 생성되고 피어 노드가 참여한 상태
      6. 복수의 피어 노드가 해당 채널에 참여하기로 되어있다면 cli의 fabric 변수를 변경해서 채널에 동일하게 가입시켜야 함(채널을 만들 필요는 없음)
3. **Channel에 ChainCode 설치**
   1. Hyperledger fabric에서 Smart Contract는 ChainCode로 명칭됨
   2. ChainCode는 채널 단위로 설치됨
   3. 현재 ChainCode 작성 지원 언어는 Go lang, Java, JavaScript, TypeScript를 지원
   4. Go lang 기준으로 ChainCode 작성 설명
      1. 체인코드 작성 폴더를 생성
      2. 폴더 내부에서 go init [package module 이름 임의로 설정] 실행

-> 패키지 관리 설정 파일 생성 명령어

* + 1. go.mod 파일 생성 확인 -> 패키지 관리 설정 파일
    2. 실제 체인코드 작성
    3. 체인코드 폴더에서 go mod vendor 실행

–> 체인코드에서 import하고 있는 모든 라이브러리를 vendor 폴더 내부로 다운로드 함

* + 1. 체인코드 설치를 위해 .tar.gz파일로 압축해야 함
       1. (cli 외부에서) docker exec [cli 컨테이너 이름] peer lifecycle chaincode package [체인코드 이름].tar.gz –path [체인코드 폴더 경로] –-lang [golang, java, javascript] –label [원하는 체인코드 라벨]
       2. 해당 명령을 통해 tar 파일로 압축된 체인코드 파일 생성
    2. 체인코드를 peer에 설치
       1. (cli 외부에서) docker exec [cli 컨테이너 이름] peer lifecycle install [생성된 체인코드 tar 파일]
       2. 설치 당시 생성된 Package ID는 다시 사용하기 때문에 환경변수로 따로 저장
       3. (cli 외부에서) docker exec cli lifecycle chaincode queryinstalled를 통해 PackageID 조회 가능
    3. 설치된 체인코드는 승인절차를 밟아야 함 (hyperledger fabric 2.x 버전 추가 절차)
       1. (cli 외부에서) docker exec [cli 컨테이너 이름] peer lifecycle chaincode approveformyorg –-channelID [채널이름] –-version 1.0 –-init-required –-package-id [PackageID] -–sequence 1 -–orderer [orderer 도메인:포트번호] –-tls –-cafile [orderer MSP tls 인증서]
       2. 복수의 조직이 채널에 가입되어 있다면 각 채널의 peer가 승인을 해야 함
    4. 설치된 체인코드의 승인절차가 완료되면 commit을 해서 체인코드 인스턴스가 docker 컨테이너 형태로 실행되게끔 해야함
       1. cli)

**트러블 슈팅**

|  |  |
| --- | --- |
| 상황 | peer에서 채널 생성 시 |
| Error log | Error: got unexpected status: FORBIDDEN -- config update for existing channel did not pass initial checks: implicit policy evaluation failed - 0 sub-policies were satisfied, but this policy requires 1 of the 'Writers' sub-policies to be satisfied: permission denied |
| 원인 | genesis.block과 channel.tx 생성 시 동일한 채널 이름 설정 |
| 해결 | Configtxgen 이용 시 genesis.block과 channel.tx에 다른 채널 이름 설정 |

|  |  |
| --- | --- |
| 상황 | Fabric sdk를 이용해 hyperledger fabric channel 연결 시 |
| Error log | Failed to get network: Failed to create new channel client: event service creation failed: could not get chConfig cache reference: QueryBlockConfig failed: QueryBlockConfig failed: queryChaincode failed: Transaction processing for endorser [localhost:7051]: Endorser Client Status Code: (2) CONNECTION\_FAILED. Description: dialing connection on target [localhost:7051]: connection is in TRANSIENT\_FAILURE  exit status 1 |
| 원인 | Fabric sdk에 자원으로 제공되는 Connection.yaml 설정의 peer 부분 조직 TLS 인증서가 틀림 |
| 해결 | TLS 인증서 확인 – 조직의 tlsca 내 인증서가 올바른 인증서 |

**참고**

|  |  |
| --- | --- |
| Alpine Liunx에서 프로그램 설치하는 법 | Apk 명령어 이용  (패키지 관리자 명령어) |
|  |  |
|  |  |