

< Hyperledger Fabric 2.0.0 개발 환경 구성 >

2020년 01월 29일 공식 release된 hyperledger fabric 2.0.0에 대한 환경 설정 방법 및 바뀐점

구 분	기 준	변 경
체인코드	일일이 설치 및 업그레이드	Decentralized governance 생성 endorsement policy와 같은 life cycle에 의해 관리
	docker container로만 관리	외부 런처도 사용 가능하도록 변경
	shim package 포함	shim package 삭제 후 dir 이동 github.com/hyperledger/fabric-chaincode-go/shim
golang	version 1.11.x 이상	version 1.13.4로 업데이트
orderer	'Solo, Kafka, Raft' 합의 알고리즘 존재	'Raft' 합의 알고리즘만 사용 가능

1. ssh : putty나 X-shell을 사용하기 위한 설치(네트워크는 '어댑터에 브릿지')

\$ sudo apt-get install openssh-server → 우분투는 기본적으로 ssh-client가 설치되어 있고 server는 미설치
server를 설치해줘야 본 서버에서 client가 설치된 서버로 원격접속 가능!!

<http://programmingskills.net/archives/315> 참고

\$ sudo service --status-all

2. curl : command url의 약자로 다양한 통신 프로토콜을 활용해 데이터를 전송하기 위해 사용

\$ sudo apt-get install -y curl → curl과 wget의 차이점(<https://brocess.tistory.com/114> 참고)

3. git

\$ sudo apt-get install -y git

4. tree

\$ sudo apt-get install -y tree

5. libtool : fabric-ca 설치 시 필요한 라이브러리로 dynamic link를 처리하기 위해 필요함

\$ sudo apt-get install -y libltdl-dev

→ <https://www.lesstif.com/pages/viewpage.action?pageId=12943542> 참고

<http://bbs.nicklib.com/application/3308> 참고(동적 라이브러리 생성에 필요)

6. net-tool

\$ sudo apt-get install -y net-tools

7. go lang : 1.13.4로 업데이트

\$ wget <https://dl.google.com/go/go1.13.4.linux-amd64.tar.gz>

\$ sudo tar -xvf go1.13.4.linux-amd64.tar.gz → x(extract, 발췌), v(verbose, 장황한), f(file) 압축푸는 옵션

\$ sudo mv go /usr/local

8. user 환경변수 설정 : .bashrc와 .profile의 차이점(<https://uroa.tistory.com/114> 참고)

\$ vi .bashrc (or .profile) → 들어가서 파일 가장 하단(shift + G)에 아래 변수 4개 작성

```
export GOROOT=/usr/local/go
export GOPATH=$HOME/go
export PATH=$PATH:$GOROOT/bin
export PATH=$PATH:$GOPATH/bin
```

\$ source .bashrc → 환경변수 4개 설정 후 활성화

\$ go env → 환경변수가 잘 설정되었는지 확인

\$ which go → 앞서 환경변수를 설정해줬기 때문에 확인해보면 /usr/local/go/bin/go에 들어있음

\$ go version → 1.13.4 확인

9. nodejs : node-sdk를 쓰기 위해서는 8.9.4 이상 또는 10.15.3 이상이어야 가능

\$ curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash - → -E(preserve-env, bash 환경보호)

(위 또는 이것)\$ wget <https://nodejs.org/dist/v10.16.1/node-v10.16.1-linux-x64.tar.xz>

\$ sudo apt-get install -y nodejs

\$ node --version → 10.18.1 나옴

\$ sudo npm install -g npm@5.6.0 → npm 버전 업그레이드(ERR! 이란 글자가 보이면 안됨)

```
project-b@projectb-VirtualBox:~$ sudo npm install -g npm@5.6.0
/usr/bin/npm -> /usr/lib/node_modules/npm/bin/npm-cli.js
/usr/bin/npx -> /usr/lib/node_modules/npm/bin/npx-cli.js
+ npm@5.6.0
added 363 packages from 147 contributors, removed 274 packages and updated 43 packages in 50.177s
```

\$ sudo npm rebuild

버전문제의 경우 재설치 해야 함

\$ sudo apt-get remove nodejs

\$ curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -

\$ sudo apt-get update

\$ sudo apt-get upgrade

\$ sudo apt-get install -y nodejs

10. python : 우분투 16.04에서는 3.5.1을 기본 제공(2.7을 사용해야 fabric 가능)

\$ sudo apt-get install -y python

\$ sudo apt-get install -y python-pip → 시간 좀 걸림

11. docker : 17.06.2 이상이어야 함

\$ curl -fsSL https://get.docker.com/ | sudo sh → 시간 오래 걸림(fail, silent, show-error, location)

\$ sudo usermod -aG docker \$(whoami) → aG(append Group), \$(whoami)에 ID입력

\$ docker --version → 17.06.2 이상 확인

12. docker-compose : 1.14.0 이상이어야 함

```
$ sudo apt-get update → 시간 오래 걸림(apt=Advanced Packaging Tool의 약자로 패키지 관리 명령어 도구)
$ sudo apt-get upgrade → 시간 오래 걸림
$ sudo curl -L https://github.com/docker/compose/releases/download/1.18.0/docker-compose-`uname
-s`-`uname -m` -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ docker images → permission denied 뜰 경우, logout 후 재접속해서 확인(아무것도 없음)
$ docker-compose --version → 1.14.0 이상 확인
```

또는

```
$ sudo apt-get install -y docker-compose
```

13. hyperledger fabric-samples

```
$ curl -sSL http://bit.ly/2ysbOFE | bash -s -- <fabric_version> <fabric-ca_version> <thirdparty_version>
→ 15분 가량 소요(ex : curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.0.0 1.4.4 0.4.18)
위의 명령은 네트워크를 설정하고 배치해야하는 모든 플랫폼 관련 바이너리를 다운로드하고
추출하는 bash 스크립트를 다운로드하고 실행(peer, orderer, fabric-ca, kafka, couch-db 등 11개)
사실상 fabric 설치 코드는 이 한 줄이며, 나머지는 docker를 사용하기 위한 환경설정용 설치
```

```
$ docker images → 11개의 이미지 별 인스턴스가 2개씩 생성됨(2.0.0과 latest / 이 각각이 fabric 동작의 핵심)
```

```
====> List out hyperledger docker images
```

hyperledger/fabric-tools	2.0	639ab50feac9	5 days ago	514MB
hyperledger/fabric-tools	2.0.0	639ab50feac9	5 days ago	514MB
hyperledger/fabric-tools	latest	639ab50feac9	5 days ago	514MB
hyperledger/fabric-peer	2.0	5f8a6b13db9f	5 days ago	57.2MB
hyperledger/fabric-peer	2.0.0	5f8a6b13db9f	5 days ago	57.2MB
hyperledger/fabric-peer	latest	5f8a6b13db9f	5 days ago	57.2MB
hyperledger/fabric-orderer	2.0	161632cc3c59	5 days ago	39.7MB
hyperledger/fabric-orderer	2.0.0	161632cc3c59	5 days ago	39.7MB
hyperledger/fabric-orderer	latest	161632cc3c59	5 days ago	39.7MB
hyperledger/fabric-ccenv	2.0	6514ca872b68	5 days ago	529MB
hyperledger/fabric-ccenv	2.0.0	6514ca872b68	5 days ago	529MB
hyperledger/fabric-ccenv	latest	6514ca872b68	5 days ago	529MB
hyperledger/fabric-baseos	2.0	50075bc26291	5 days ago	6.9MB
hyperledger/fabric-baseos	2.0.0	50075bc26291	5 days ago	6.9MB
hyperledger/fabric-baseos	latest	50075bc26291	5 days ago	6.9MB
hyperledger/fabric-javaenv	2.0	ac433f4353e4	10 days ago	507MB
hyperledger/fabric-javaenv	2.0.0	ac433f4353e4	10 days ago	507MB
hyperledger/fabric-javaenv	latest	ac433f4353e4	10 days ago	507MB
hyperledger/fabric-nodeenv	2.0	c7fe428889ec	11 days ago	274MB
hyperledger/fabric-nodeenv	2.0.0	c7fe428889ec	11 days ago	274MB
hyperledger/fabric-nodeenv	latest	c7fe428889ec	11 days ago	274MB
hyperledger/fabric-ca	1.4	62a60c5459ae	2 months ago	150MB
hyperledger/fabric-ca	1.4.4	62a60c5459ae	2 months ago	150MB
hyperledger/fabric-ca	latest	62a60c5459ae	2 months ago	150MB
hyperledger/fabric-zookeeper	0.4	ede9389347db	3 months ago	276MB
hyperledger/fabric-zookeeper	0.4.18	ede9389347db	3 months ago	276MB
hyperledger/fabric-zookeeper	latest	ede9389347db	3 months ago	276MB
hyperledger/fabric-kafka	0.4	caaa0474ef2	3 months ago	270MB
hyperledger/fabric-kafka	0.4.18	caaa0474ef2	3 months ago	270MB
hyperledger/fabric-kafka	latest	caaa0474ef2	3 months ago	270MB
hyperledger/fabric-couchdb	0.4	d369d4eaa0fd	3 months ago	261MB
hyperledger/fabric-couchdb	0.4.18	d369d4eaa0fd	3 months ago	261MB
hyperledger/fabric-couchdb	latest	d369d4eaa0fd	3 months ago	261MB

14. user 환경변수 설정 : .bashrc와 .profile의 차이점(<https://uoa.tistory.com/114> 참고)

```
$ vi .bashrc (or .profile) → 들어가서 파일 가장 하단(shift + G)에 아래 변수 작성
```

```
export FABRIC_HOME=$GOPATH/src/github.com/hyperledger/fabric
export PATH=$PATH:$GOPATH/src/github.com/hyperledger/fabric/build/bin/
export PATH=$PATH:$GOPATH/src/github.com/hyperledger/fabric-ca/bin/
```

```
$ source .bashrc → 환경변수 3개 설정 후 활성화
```

15. fabric 2.0.0

```
$ mkdir -p $GOPATH/src/github.com/hyperledger → home에서 실행(path 옵션을 통해 4개 dir 한 번에 생성)
$ cd $GOPATH/src/github.com/hyperledger → $GOPATH는 위에 환경변수 설정해둠($HOME/gopath)
$ git clone -b release-2.0 https://github.com/hyperledger/fabric → branch를 명시하지 않으면 master로 받아짐
$ git branch → *release-2.0 확인
$ cd $GOPATH/src/github.com/hyperledger/fabric
$ make → unit-test 이후 반복적으로 코드들 나오면 ctrl+c를 통해 종료(약 1시간(?) 소요 so 오래 걸림)
→ make는 프로그램 그룹을 유지하는데 필요한 유틸리티로 자동 compile 해주는 역할
소스파일로 이루어진 것들을 컴파일 해 실행파일로 만들어 줌(https://maeuminpaper.tistory.com/83 참고)
https://wiki.kldp.org/KoreanDoc/html/gcc\_and\_make/gcc\_and\_make-3.html 참고
gcc(GNU Compiler Collection) 명령어를 통해 make가 만들어짐
```

16. fabric-ca 1.4.4

```
$ cd $GOPATH/src/github.com/hyperledger → $GOPATH는 위에 환경변수 설정해둠($HOME/gopath)
$ git clone -b v1.4.4 https://github.com/hyperledger/fabric-ca
$ cd $GOPATH/src/github.com/hyperledger/fabric-ca
$ make fabric-ca-server → 아래 결과화면 참고
$ make fabric-ca-client → 아래 결과화면 참고
```

```
project-b@projectb-VirtualBox:~/gopath/src/github.com/hyperledger/fabric-ca$ make fabric-ca-server
Building fabric-ca-server in bin directory ...
Built bin/fabric-ca-server
project-b@projectb-VirtualBox:~/gopath/src/github.com/hyperledger/fabric-ca$ make fabric-ca-client
Building fabric-ca-client in bin directory ...
Built bin/fabric-ca-client
```

17. Window 엑스트라 : Windows 7에서 개발하는 경우에만 해당

```
$ git config --global core.autocrlf false
$ git config --global core.longpaths true
위 두 명령어 실행 후
$ git config --get core.autocrlf → false 확인
$ git config --get core.longpaths → true 확인
$ sudo npm install --global windows-build-tools
→ node.js의 자유로운 활용을 위한 Visual Studio C++ 빌드도구 설치(linux라 오류 발생)
$ sudo npm install --global grpc --unsafe-perm → --unsafe-perm 옵션을 통해 npm이 root 계정으로 실행하도록 지정
```


18. 최종 확인(https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html 참고)

```
$ cd fabric-samples/test-network
```

```
$ ./network.sh up -ca → 명령어 실행 시 docker images 6개 생성
```

```
Generate CCP files for Org1 and Org2
/home/kismi/fabric-samples/test-network/bin/configtxgen
##### Generating Orderer Genesis block #####
+ configtxgen -profile TwoOrgsOrdererGenesis -channelID system-channel -outputBlock ./system-genesis-block/genesis.block
2020-02-04 07:21:53.242 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-02-04 07:21:53.273 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2020-02-04 07:21:53.274 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_i
nflight_blocks:5 snapshot_interval_size:16777216
2020-02-04 07:21:53.274 UTC [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /home/kismi/fabric-samples/test-network/configtx/configtx.yaml
2020-02-04 07:21:53.278 UTC [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2020-02-04 07:21:53.279 UTC [common.tools.configtxgen] doOutputBlock -> INFO 006 Writing genesis block
+ res=0
+ set +x
Creating network "net_test" with the default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_orderer.example.com" with default driver
WARNING: Found orphan containers (ca_orderer, ca_org1, ca_org2) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clea
n it up.
Creating peer0.org1.example.com ... done
Creating peer0.org1.example.com ...
Creating peer0.org2.example.com ...
Creating orderer.example.com ...
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS                               NAMES
8f31b397a31c       hyperledger/fabric-peer:latest      "peer node start"       6 seconds ago      Up Less than a second    0.0.0.0:7051->7051/tcp        peer0.org1.example.com
df51df63b61b       hyperledger/fabric-orderer:latest  "orderer"               6 seconds ago      Up 1 second           0.0.0.0:7050->7050/tcp        orderer.example.com
f626b39bd2db       hyperledger/fabric-peer:latest      "peer node start"       6 seconds ago      Up 1 second           7051/tcp, 0.0.0.0:9051->9051/tcp peer0.org2.example.com
86fcdedd7037       hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se-" 29 seconds ago     Up 24 seconds          7054/tcp, 0.0.0.0:9054->9054/tcp ca_orderer
cb75fd9c1ddc       hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se-" 29 seconds ago     Up 25 seconds          0.0.0.0:7054->7054/tcp        ca_org1
949cf6df08e        hyperledger/fabric-ca:latest        "sh -c 'fabric-ca-se-" 29 seconds ago     Up 23 seconds          7054/tcp, 0.0.0.0:8054->8054/tcp ca_org2
```

만약 안 될 경우?

```
$ ./network.sh down 또는 아래 코드
```

```
$ docker stop $(docker ps -qa) && docker rm $(docker ps -qa)
```

```
$ ./network.sh up → 컨테이너 간 꼬여서 error 발생한 것이기 때문에 지우고 다시 실행하면 이상 무!
```

```
$ ./network.sh createChannel → genesis.block 생성 및 channel join 후 successfully 확인
```

```
$ ./network.sh deployCC → channel에서 chaincode 설치 및 배포(잘 되어 마지막에 successfully 확인)
```

```
===== Invoke transaction successful on peer0.org1 peer0.org2 on channel 'mychannel' =====
Querying chaincode on peer0.org1...
===== Querying on peer0.org1 on channel 'mychannel'... =====
Attempting to Query peer0.org1 ...1579498896 secs
++ peer chaincode query -C mychannel -n fabcar -c '{"Args":["queryAllCars"]}'
++ res=0
++ set +x

[{"Key":"CAR0","Record":{"make":"Toyota","model":"Prius","colour":"blue","owner":"Tomoko"}}, {"Key":"CAR1","Record":{"make":"Ford","model":"Mustang","colour":"red","owner":"Brad"}}, {"Key":"CAR2","Record":{"make":"Hyundai","model":"Tucson","colour":"green","owner":"Jin Soo"}}, {"Key":"CAR3","Record":{"make":"Volkswagen","model":"Passat","colour":"yellow","owner":"Max"}}, {"Key":"CAR4","Record":{"make":"Tesla","model":"S","colour":"black","owner":"Adriana"}}, {"Key":"CAR5","Record":{"make":"Peugeot","model":"205","colour":"purple","owner":"Michel"}}, {"Key":"CAR6","Record":{"make":"Chery","model":"S22L","colour":"white","owner":"Aarav"}}, {"Key":"CAR7","Record":{"make":"Fiat","model":"Punto","colour":"violet","owner":"Pari"}}, {"Key":"CAR8","Record":{"make":"Tata","model":"Nano","colour":"indigo","owner":"Valeria"}}, {"Key":"CAR9","Record":{"make":"Holden","model":"Barina","colour":"brown","owner":"Shotaro"}}]
===== Query successful on peer0.org1 on channel 'mychannel' =====
```