< 멀티호스트 설정 후 blockchain-explorer로 확인하기 >

https://github.com/wahabjawed/Build-Multi-Host-Network-Hyperledger → 공식 github 사이트
https://medium.com/@wahabjawed/hyperledger-fabric-on-multiple-hosts-a33b08ef24f → 매뉴얼 참고 사이트

1. Build-Multi-Host-Network-Hyperledger 다운받기

```
$ git clone https://github.com/wahabjawed/Build-Multi-Host-Network-Hyperledger.git
```

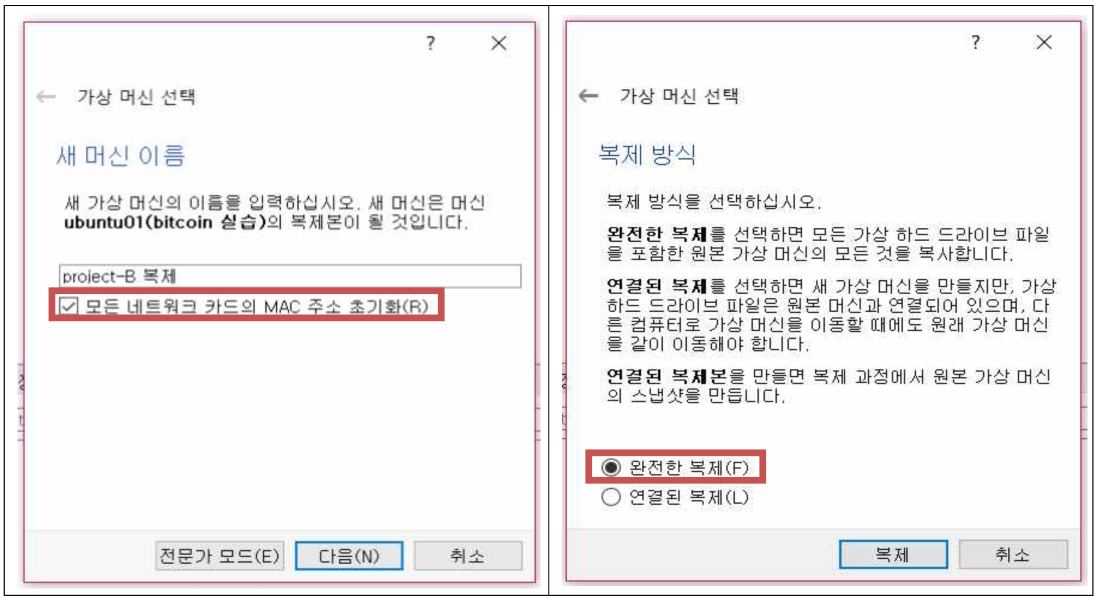
2. 멀티호스트 네트워크 구성요소 설정

```
$ cd Build-Multi-Host-Network-Hyperledger
$ ./bmhn generate
```

→ channel-artifacts 아래 genesis block, anchor peer, channel 생성
crypto-config 아래 orderer, peer 관련 인증서 파일들 생성

3. PC2 생성(VM 복제)

- 위 과정을 통해 PC1의 기본설정을 마쳤으므로 PC2 생성(local에서 진행하므로 VM 이용)
- VM 끄고 → 마우스 우클릭 → 복제 → 아래 옵션 적용 후 '복제'



← 가상 머신 선택

새 머신 이름

새 가상 머신의 이름을 입력하십시오. 새 머신은 머신 **ubuntu01(bitcoin 실습)**의 복제본이 될 것입니다.

project-B 복제

[✓] 모든 네트워크 카드의 MAC 주소 초기화(R)

← 가상 머신 선택

복제 방식

복제 방식을 선택하십시오.

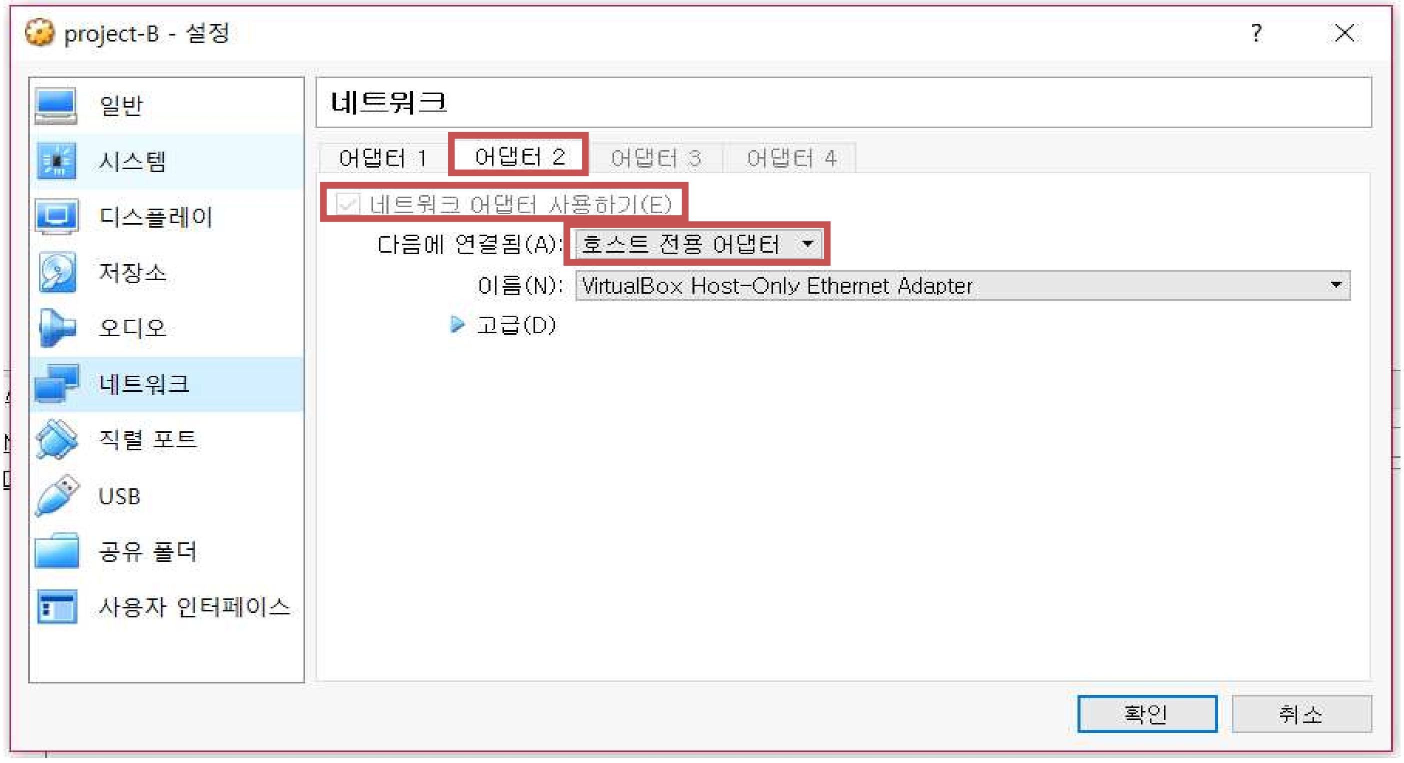
완전한 복제를 선택하면 모든 가상 하드 드라이브 파일을 포함한 원본 가상 머신의 모든 것을 복사합니다.

연결된 복제를 선택하면 새 가상 머신을 만들지만, 가상 하드 드라이브 파일은 원본 머신과 연결되어 있으며, 다른 컴퓨터로 가상 머신을 이동할 때에도 원래 가상 머신을 같이 이동해야 합니다.

연결된 복제본을 만들면 복제 과정에서 원본 가상 머신의 스냅샷을 만듭니다.

[● 완전한 복제(F)] ○ 연결된 복제(L)

- 생성된 VM 별 네트워크 어댑터 설정(서로 통신을 위해 '어탭터2'를 아래와 같이 설정)



project-B - 설정

네트워크

어댑터 1 | [어댑터 2] | 어댑터 3 | 어댑터 4

[✓] 네트워크 어댑터 사용하기(E)

다음에 연결됨(A): 호스트 전용 어댑터

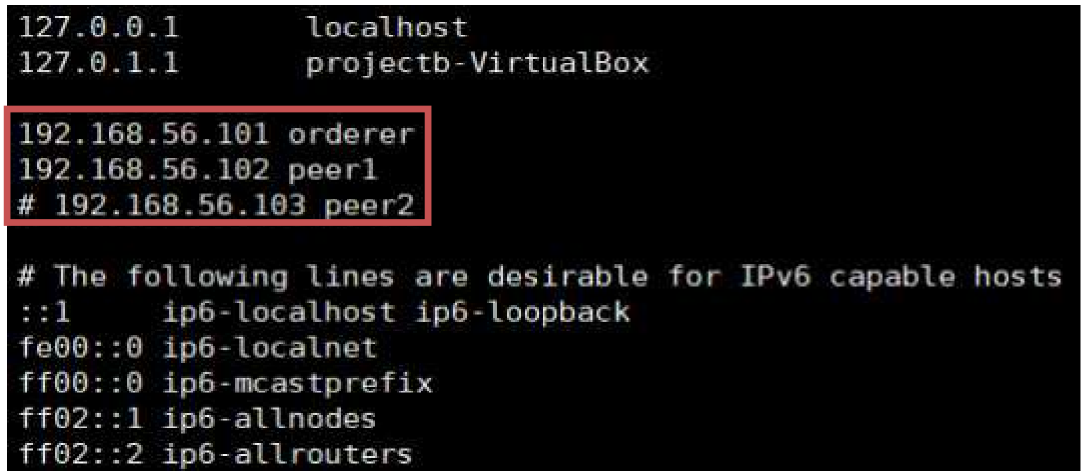
이름(N): VirtualBox Host-Only Ethernet Adapter

4. 고정IP 설정(서버모드는 http://blog.hkwon.me/virtualbox-hoseuteu-jeonyong-eodaebteo-seoljeong-2/ 참고)

$ sudo vi /etc/hosts → 편집기를 이용해 /etc/hosts 파일 열기

```
192.168.56.101  orderer
192.168.56.102  peer1
#  192.168.56.103  peer2
```

위와 같이 코드 추가(해당 IP별 name과 함께 할당하는 작업)

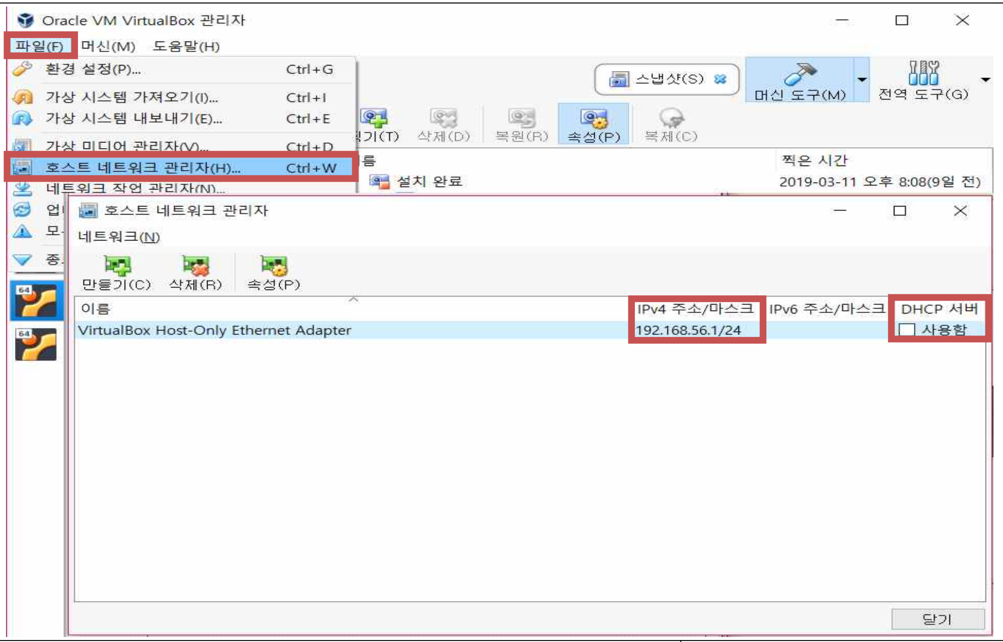


```
127.0.0.1       localhost
127.0.1.1       projectb-VirtualBox

192.168.56.101 orderer
192.168.56.102 peer1
# 192.168.56.103 peer2

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

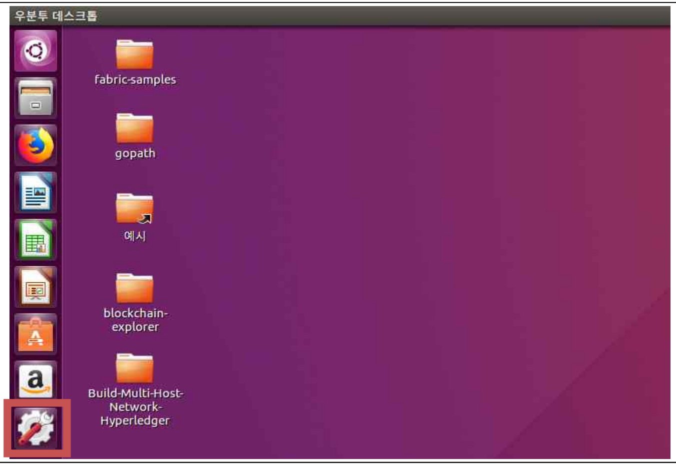
- 아래 과정 VM 머신 PC1과 PC2 각각 실행
  설정 → 네트워크 → 아무것도 없는 '유선' 선택 → 옵션 → 이더넷 → 'enp0s8'로 장치 선택
  → 저장 → IPv4설정 → 방식 '수동' 선택 → 주소 및 넷마스크 추가(/etc/hosts에서 설정한대로 작성)
  → 저장 → 옵션의 IPv4 주소가 설정한대로 바뀐 것 확인!

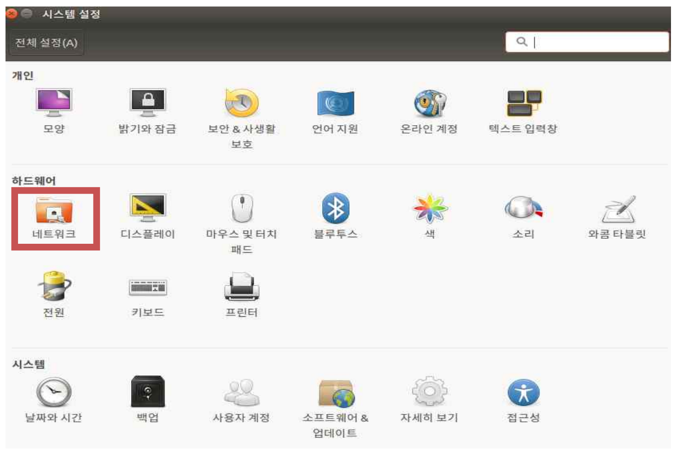

Oracle VM VirtualBox 관리자
파일(F)
머신(M)
도움말(H)
환경 설정(P)... Ctrl+G
가상 시스템 가져오기(I)... Ctrl+I
가상 시스템 내보내기(E)... Ctrl+E
가상 미디어 관리자(V)... Ctrl+D
호스트 네트워크 관리자(H)... Ctrl+W
네트워크 작업 관리자(N)...
스냅샷(S)
머신 도구(M)
전역 도구(G)
삭제(D)
복원(R)
속성(P)
복제(C)
찍은 시간
설치 완료
2019-03-11 오후 8:08(9일 전)
호스트 네트워크 관리자
네트워크(N)
만들기(C)
삭제(R)
속성(P)
닫기

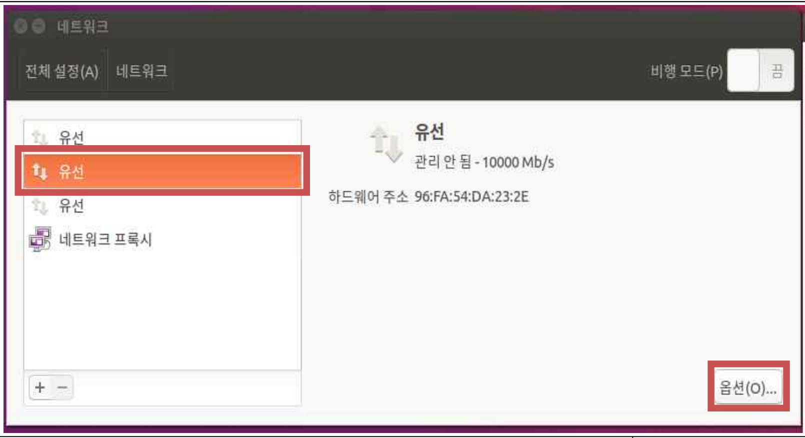
이름	IPv4 주소/마스크	IPv6 주소/마스크	DHCP 서버
VirtualBox Host-Only Ethernet Adapter	192.168.56.1/24		☐ 사용함

· 호스트 네트워크 관리자에 들어가서 **IPv4의 3번째 자리** 확인 = 56

· DHCP 서버의 사용은 사용함 **체크해제!**



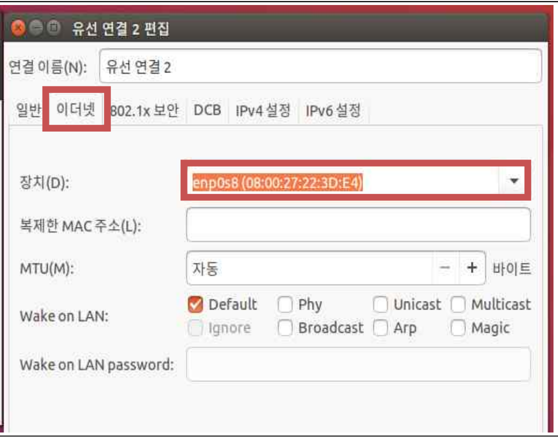



네트워크

전체 설정(A) | 네트워크 | 비행 모드(P) 끔

- 유선
- 유선
- 유선
- 네트워크 프록시

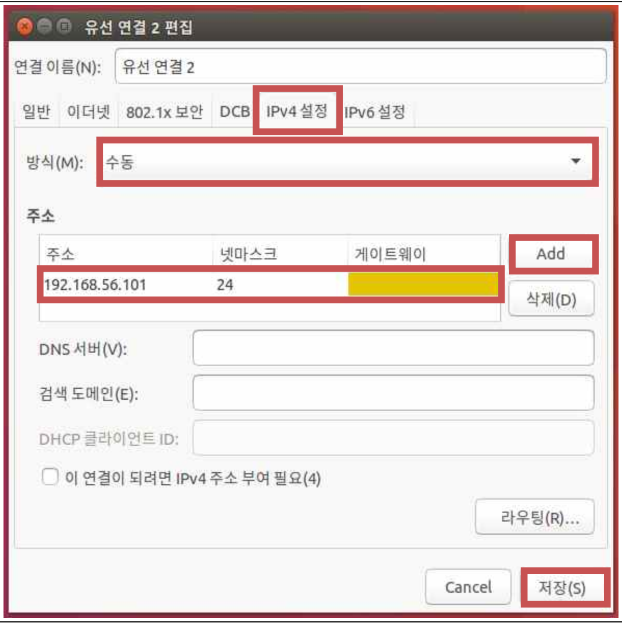
유선
관리 안 됨 - 10000 Mb/s
하드웨어 주소 96:FA:54:DA:23:2E


유선 연결 2 편집

연결 이름(N): 유선 연결 2

일반 | 이더넷 | 802.1x 보안 | DCB | IPv4 설정 | IPv6 설정

장치(D): enp0s8 (08:00:27:22:3D:E4)
복제한 MAC 주소(L):
MTU(M): 자동 − + 바이트
Wake on LAN: [✓ Default] Phy Unicast Multicast Ignore Broadcast Arp Magic
Wake on LAN password:


유선 연결 2 편집

연결 이름(N): 유선 연결 2

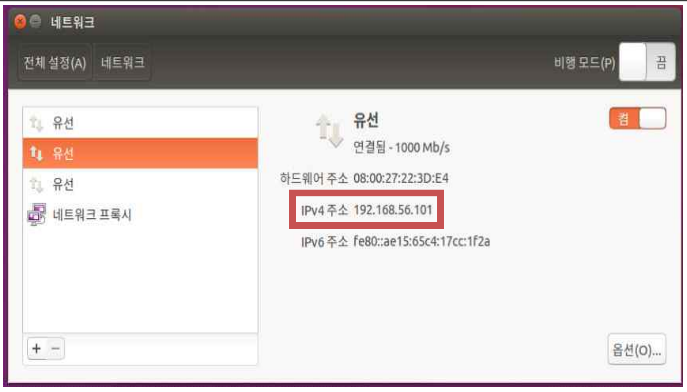
일반 | 이더넷 | 802.1x 보안 | DCB | IPv4 설정 | IPv6 설정

방식(M): 수동

주소

주소	넷마스크	게이트웨이
192.168.56.101	24	🟨

DNS 서버(V):
검색 도메인(E):
DHCP 클라이언트 ID:
☐ 이 연결이 되려면 IPv4 주소 부여 필요(4)


네트워크

전체 설정(A) | 네트워크 | 비행 모드(P) 끔

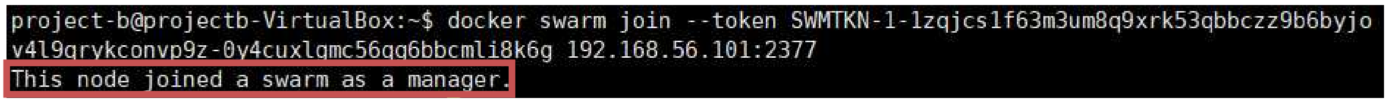
- 유선
- 유선
- 유선
- 네트워크 프록시

유선 켬
연결됨 - 1000 Mb/s
하드웨어 주소 08:00:27:22:3D:E4
IPv4 주소 192.168.56.101
IPv6 주소 fe80::ae15:65c4:17cc:1f2a

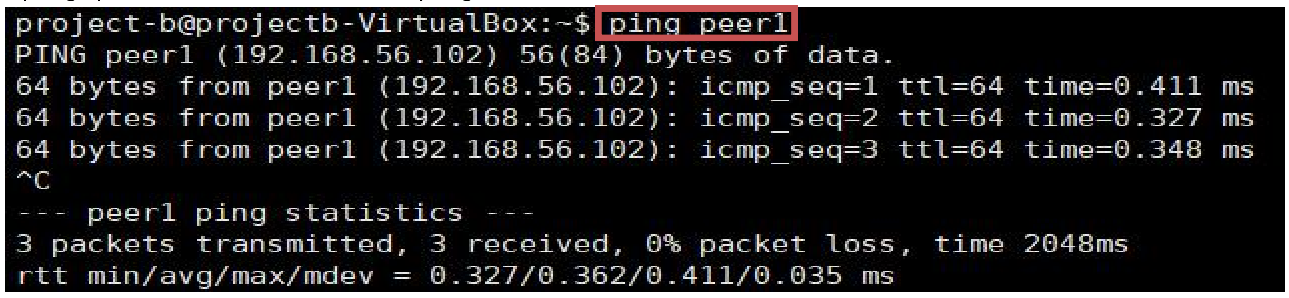
VirtualBox의 **네트워크 디바이스 이름**이 enp0s3/8 2개
enp0s8 : 인터넷과 외부에서 접속할 수 있는
유동 IP의 통로 역할
enp0s3 : 관리용 네트워크로 대시보드 및 API에 접속 가능
http://blog.hkwon.me/virtualbox-hoseuteu-jeonyong-eodaebteo-seoljeong-2/ 참고

5. docker swarm 작업(https://subicura.com/2017/02/25/container-orchestration-with-docker-swarm.html 참고)
① PC1 = 192.168.56.101
$ docker swarm init --advertise-addr 192.168.56.101 → 해당 IP로 swarm
($ docker swarm leave --force → 이전 기록이 있다면 삭제)
$ docker swarm join-token manager → 매니저(관리자)로 join(하면 아래 복잡한 코드가 뜸)

② PC2 = 192.168.56.102
$ docker swarm join –token SWMTKN-1-1zqjcs1f63m3um8q9xrk53qbbczz9b6byjov4l9qrykconvp9z-0y4cuxlgmc56qg6bbcmli8k6g 192.168.56.101:2377
→ 위에 join-token 했을 때 나오는 코드 그대로 복사해서 붙여넣기 하는 것(joined 확인)


```
project-b@projectb-VirtualBox:~$ docker swarm join --token SWMTKN-1-1zqjcs1f63m3um8q9xrk53qbbczz9b6byjo
v4l9qrykconvp9z-0y4cuxlgmc56qg6bbcmli8k6g 192.168.56.101:2377
This node joined a swarm as a manager.
```

③ 잘 연결되었는지 확인하기 위해 각 PC별 ping을 날려보기(제대로 날라가야 join 된 것)
$ ping peer1(PC1에서) 또는 $ ping orderer(PC2에서)


```
project-b@projectb-VirtualBox:~$ ping peer1
PING peer1 (192.168.56.102) 56(84) bytes of data.
64 bytes from peer1 (192.168.56.102): icmp_seq=1 ttl=64 time=0.411 ms
64 bytes from peer1 (192.168.56.102): icmp_seq=2 ttl=64 time=0.327 ms
64 bytes from peer1 (192.168.56.102): icmp_seq=3 ttl=64 time=0.348 ms
^C
--- peer1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.327/0.362/0.411/0.035 ms
```

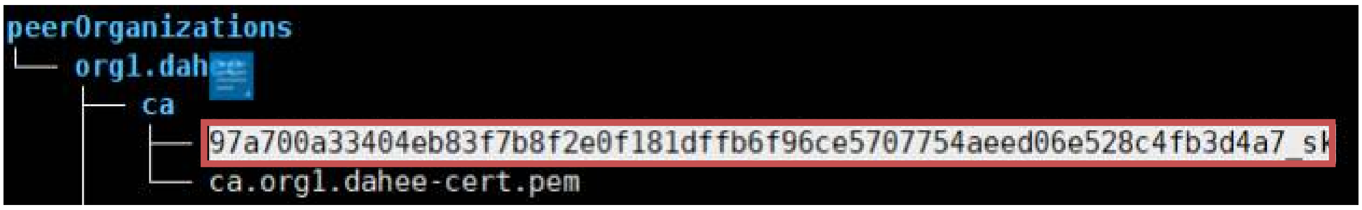
④ 최종적으로 network 생성하기(PC1에서)
$ docker network create --attachable --driver overlay my-net → 네트워크 생성 완료

6. PC1에 CA, Orderer, peer0, couch0 구동

① CA-server

```
$ docker run --rm -it --network="my-net" --name ca.example.com -p 7054:7054 -e
FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server -e FABRIC_CA_SERVER_CA_NAME=ca.example.com
- e
FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem
– e
FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/97a700a33404eb83f7b8f2e0f181
dffb6f96ce5707754aeed06e528c4fb3d4a7_sk -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=hyp-net hyperledger/fabric-ca sh -c 'fabric-ca-server
start -b admin:adminpw -d'
```

```
$ cd ~/Build-Multi-Host-Network-Hyperledger
$ tree crypto-config → 해서 나오는 peer 아래 ca의 key를 위의 명령어에 써 줄 것!!(파랑글씨)
```



```
peerOrganizations
└── org1.dah
    ├── ca
    │   ├── 97a700a33404eb83f7b8f2e0f181dffb6f96ce5707754aeed06e528c4fb3d4a7_sk
    │   └── ca.org1.dahee-cert.pem
```

② Orderer

```
$ docker run --rm -it --network="my-net" --name orderer.example.com -p 7050:7050 -e
ORDERER_GENERAL_LOGLEVEL=debug -e ORDERER_GENERAL_LISTENADDRESS=0.0.0.0 -e
ORDERER_GENERAL_LISTENPORT=7050 -e ORDERER_GENERAL_GENESISMETHOD=file -e
ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block -e
ORDERER_GENERAL_LOCALMSPID=OrdererMSP -e
ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp -e
ORDERER_GENERAL_TLS_ENABLED=false -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net
-v $(pwd)/channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block -v
$(pwd)/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperled
ger/orderer/msp -w /opt/gopath/src/github.com/hyperledger/fabric hyperledger/fabric-orderer orderer
```

③ CouchDB0

```
$ docker run --rm -it --network="my-net" --name couchdb0 -p 5984:5984 -e COUCHDB_USER= -e
COUCHDB_PASSWORD= -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net
hyperledger/fabric-couchdb
```

④ Peer0

```
$ docker run --rm -it --link orderer.example.com:orderer.example.com --network="my-net" --name
peer0.org1.example.com -p 8051:7051 -p 8053:7053 -e CORE_LEDGER_STATE_STATEDATABASE=CouchDB
-e CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984 -e
CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= -e
CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= -e CORE_PEER_ADDRESSAUTODETECT=true -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e FABRIC_LOGGING_SPEC=DEBUG -e
CORE_PEER_NETWORKID=peer0.org1.example.com -e CORE_NEXT=true -e
CORE_PEER_ENDORSER_ENABLED=true -e CORE_PEER_ID=peer0.org1.example.com -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:7050 -e
CORE_PEER_GOSSIP_IGNORESECURITY=true -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051 -e
```

CORE_PEER_TLS_ENABLED=false -e CORE_PEER_GOSSIP_USELEADERELECTION=false -e CORE_PEER_GOSSIP_ORGLEADER=true -e CORE_PEER_LOCALMSPID=Org1MSP -v /var/run/:/host/var/run/ -v $(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-peer peer node start

* CORE_LOGGING_LEVEL은 지원이 되지 않으므로, FABRIC_LOGGING_SPEC로 바꿔줘야 함.

7. PC2에 peer1, couch1, cli 구동

① CouchDB1

$ docker run --rm -it --network="my-net" --name couchdb1 -p 6984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net hyperledger/fabric-couchdb

② Peer1

$ docker run --rm -it --network="my-net" --link orderer.example.com:orderer.example.com --link peer0.org1.example.com:peer0.org1.example.com --name peer1.org1.example.com -p 9051:7051 -p 9053:7053 -e CORE_LEDGER_STATE_STATEDATABASE=CouchDB -e CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984 -e CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= -e CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= -e CORE_PEER_ADDRESSAUTODETECT=true -e CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e FABRIC_LOGGING_SPEC=DEBUG -e CORE_PEER_NETWORKID=peer1.org1.example.com -e CORE_NEXT=true -e CORE_PEER_ENDORSER_ENABLED=true -e CORE_PEER_ID=peer1.org1.example.com -e CORE_PEER_PROFILE_ENABLED=true -e CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:7050 -e CORE_PEER_GOSSIP_ORGLEADER=true -e CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:7051 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -e CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051 -e CORE_PEER_GOSSIP_USELEADERELECTION=false -e CORE_PEER_TLS_ENABLED=false -v /var/run/:/host/var/run/ -v $(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp:/etc/hyperledger/fabric/msp -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-peer peer node start

* CORE_LOGGING_LEVEL은 지원이 되지 않으므로, FABRIC_LOGGING_SPEC로 바꿔줘야 함.

③ cli

$ docker run --rm -it --network="my-net" --name cli --link orderer.example.com:orderer.example.com --link peer0.org1.example.com:peer0.org1.example.com --link peer1.org1.example.com:peer1.org1.example.com -p 12051:7051 -p 12053:7053 -e GOPATH=/opt/gopath -e CORE_PEER_LOCALMSPID=Org1MSP -e CORE_PEER_TLS_ENABLED=false -e CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e FABRIC_LOGGING_SPEC=DEBUG -e CORE_PEER_ID=cli -e CORE_PEER_ADDRESS=peer0.org1.example.com:7051 -e CORE_PEER_NETWORKID=cli -e CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=my-net -v /var/run/:/host/var/run/ -v $(pwd)/chaincode/:/opt/gopath/src/github.com/hyperledger/fabric/examples/chaincode/go -v $(pwd)/crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ -v

$(pwd)/scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/ -v $(pwd)/channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts -w /opt/gopath/src/github.com/hyperledger/fabric/peer hyperledger/fabric-tools /bin/bash

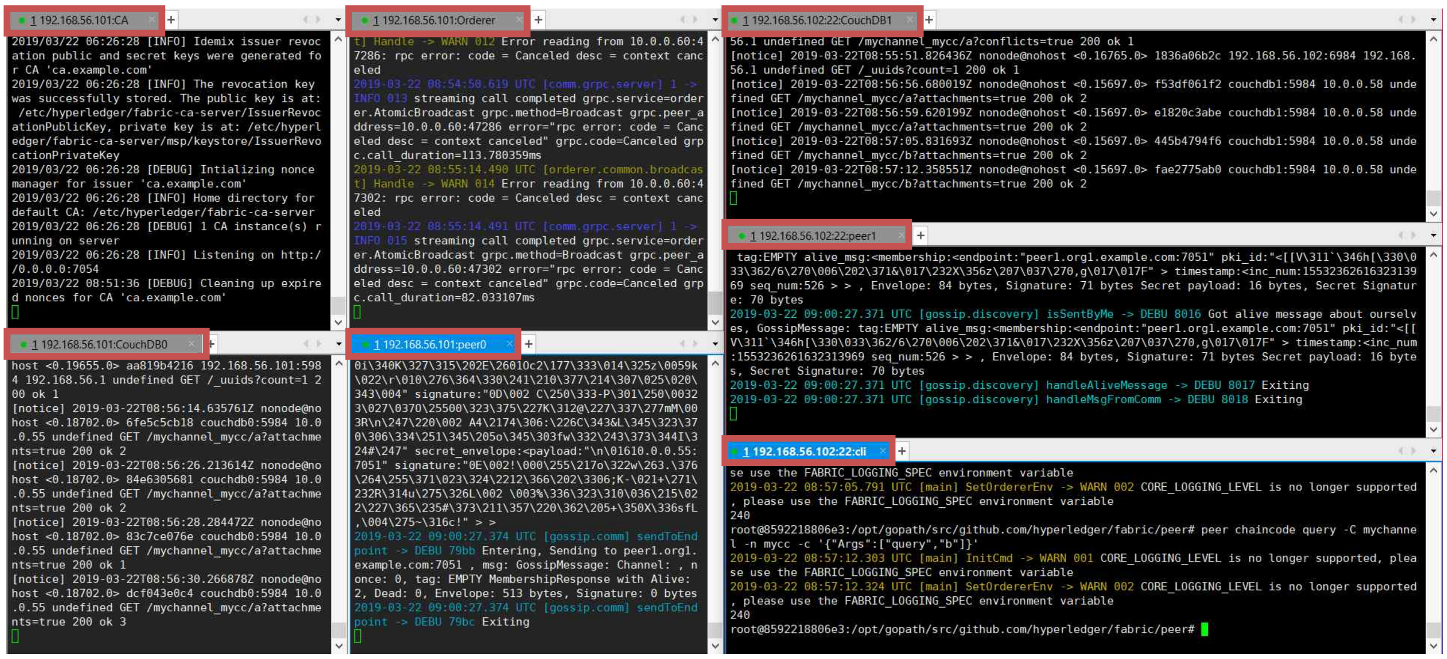
* CORE_LOGGING_LEVEL은 지원이 되지 않으므로, FABRIC_LOGGING_SPEC로 바꿔줘야 함.

* 이런 창이 뜸

```
root@51910d764fee:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

# ./scripts/script.sh →을 통해 cli 스크립트를 실행시켜 up 시키기!

* 이후 설치된 chaincode를 instantiate 해주면 완료!!(이후 invoke, query 등 날려보기)



```
2019/03/22 06:26:28 [INFO] Idemix issuer revoc
ation public and secret keys were generated fo
r CA 'ca.example.com'
2019/03/22 06:26:28 [INFO] The revocation key
was successfully stored. The public key is at:
 /etc/hyperledger/fabric-ca-server/IssuerRevoc
ationPublicKey, private key is at: /etc/hyperl
edger/fabric-ca-server/msp/keystore/IssuerRevo
cationPrivateKey
2019/03/22 06:26:28 [DEBUG] Intializing nonce
manager for issuer 'ca.example.com'
2019/03/22 06:26:28 [INFO] Home directory for
default CA: /etc/hyperledger/fabric-ca-server
2019/03/22 06:26:28 [DEBUG] 1 CA instance(s) r
unning on server
2019/03/22 06:26:28 [INFO] Listening on http:/
/0.0.0.0:7054
2019/03/22 08:51:36 [DEBUG] Cleaning up expire
d nonces for CA 'ca.example.com'
```

```
t] Handle -> WARN 012 Error reading from 10.0.0.60:4
7286: rpc error: code = Canceled desc = context canc
eled
2019-03-22 08:54:50.619 UTC [comm.grpc.server] 1 ->
INFO 013 streaming call completed grpc.service=order
er.AtomicBroadcast grpc.method=Broadcast grpc.peer_a
ddress=10.0.0.60:47286 error="rpc error: code = Canc
eled desc = context canceled" grpc.code=Canceled grp
c.call_duration=113.780359ms
2019-03-22 08:55:14.490 UTC [orderer.common.broadcas
t] Handle -> WARN 014 Error reading from 10.0.0.60:4
7302: rpc error: code = Canceled desc = context canc
eled
2019-03-22 08:55:14.491 UTC [comm.grpc.server] 1 ->
INFO 015 streaming call completed grpc.service=order
er.AtomicBroadcast grpc.method=Broadcast grpc.peer_a
ddress=10.0.0.60:47302 error="rpc error: code = Canc
eled desc = context canceled" grpc.code=Canceled grp
c.call_duration=82.033107ms
```

```
56.1 undefined GET /mychannel_mycc/a?conflicts=true 200 ok 1
[notice] 2019-03-22T08:55:51.826436Z nonode@nohost <0.16765.0> 1836a06b2c 192.168.56.102:6984 192.168.
56.1 undefined GET /_uuids?count=1 200 ok 1
[notice] 2019-03-22T08:56:56.680019Z nonode@nohost <0.15697.0> f53df061f2 couchdb1:5984 10.0.0.58 unde
fined GET /mychannel_mycc/a?attachments=true 200 ok 2
[notice] 2019-03-22T08:56:59.620199Z nonode@nohost <0.15697.0> e1820c3abe couchdb1:5984 10.0.0.58 unde
fined GET /mychannel_mycc/a?attachments=true 200 ok 2
[notice] 2019-03-22T08:57:05.831693Z nonode@nohost <0.15697.0> 445b4794f6 couchdb1:5984 10.0.0.58 unde
fined GET /mychannel_mycc/b?attachments=true 200 ok 2
[notice] 2019-03-22T08:57:12.358551Z nonode@nohost <0.15697.0> fae2775ab0 couchdb1:5984 10.0.0.58 unde
fined GET /mychannel_mycc/b?attachments=true 200 ok 2
```

```
 tag:EMPTY alive_msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"<[[V\311`\346h[\330\0
33\362/6\270\006\202\371&\017\232X\356z\207\037\270,g\017\017F" > timestamp:<inc_num:15532362616323139
69 seq_num:526 > > , Envelope: 84 bytes, Signature: 71 bytes Secret payload: 16 bytes, Secret Signatur
e: 70 bytes
2019-03-22 09:00:27.371 UTC [gossip.discovery] isSentByMe -> DEBU 8016 Got alive message about ourselv
es, GossipMessage: tag:EMPTY alive_msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"<[[
V\311`\346h[\330\033\362/6\270\006\202\371&\017\232X\356z\207\037\270,g\017\017F" > timestamp:<inc_num
:15532362616323139 69 seq_num:526 > > , Envelope: 84 bytes, Signature: 71 bytes Secret payload: 16 byte
s, Secret Signature: 70 bytes
2019-03-22 09:00:27.371 UTC [gossip.discovery] handleAliveMessage -> DEBU 8017 Exiting
2019-03-22 09:00:27.371 UTC [gossip.discovery] handleMsgFromComm -> DEBU 8018 Exiting
```

```
host <0.19655.0> aa819b4216 192.168.56.101:598
4 192.168.56.1 undefined GET /_uuids?count=1 2
00 ok 1
[notice] 2019-03-22T08:56:14.635761Z nonode@no
host <0.18702.0> 6fe5c5cb18 couchdb0:5984 10.0
.0.55 undefined GET /mychannel_mycc/a?attachme
nts=true 200 ok 2
[notice] 2019-03-22T08:56:26.213614Z nonode@no
host <0.18702.0> 84e6305681 couchdb0:5984 10.0
.0.55 undefined GET /mychannel_mycc/a?attachme
nts=true 200 ok 2
[notice] 2019-03-22T08:56:28.284472Z nonode@no
host <0.18702.0> 83c7ce076e couchdb0:5984 10.0
.0.55 undefined GET /mychannel_mycc/a?attachme
nts=true 200 ok 1
[notice] 2019-03-22T08:56:30.266878Z nonode@no
host <0.18702.0> dcf043e0c4 couchdb0:5984 10.0
.0.55 undefined GET /mychannel_mycc/a?attachme
nts=true 200 ok 3
```

```
0i\340K\327\315\202E\26010c2\177\333\014\325z\0059k
\022\r\010\276\364\330\241\210\377\214\307\025\020\
343\004" signature:"0D\002 C\250\333-P\301\250\0032
3\027\0370\25500\323\375\227K\312@\227\337\277mM\00
3R\n\247\220\002 A4\2174\306:\226C\343&L\345\323\37
0\306\334\251\345\205o\345\303fw\332\243\373\344I\3
24#\247" secret_envelope:<payload:"\n\01610.0.0.55:
7051" signature:"0E\002!\000\255\217o\322w\263.\376
\264\255\371\023\324\2212\366\202\3306;K-\021+\271\
232R\314u\275\326L\002 \003%\336\323\310\036\215\02
2\227\365\235#\373\211\357\220\362\205+\350X\336sfL
,\004\275~\316c!" > >
2019-03-22 09:00:27.374 UTC [gossip.comm] sendToEnd
point -> DEBU 79bb Entering, Sending to peer1.org1.
example.com:7051 , msg: GossipMessage: Channel: , n
once: 0, tag: EMPTY MembershipResponse with Alive:
2, Dead: 0, Envelope: 513 bytes, Signature: 0 bytes
2019-03-22 09:00:27.374 UTC [gossip.comm] sendToEnd
point -> DEBU 79bc Exiting
```

```
se use the FABRIC_LOGGING_SPEC environment variable
2019-03-22 08:57:05.791 UTC [main] SetOrdererEnv -> WARN 002 CORE_LOGGING_LEVEL is no longer supported
, please use the FABRIC_LOGGING_SPEC environment variable
240
root@8592218806e3:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychanne
l -n mycc -c '{"Args":["query","b"]}'
2019-03-22 08:57:12.303 UTC [main] InitCmd -> WARN 001 CORE_LOGGING_LEVEL is no longer supported, plea
se use the FABRIC_LOGGING_SPEC environment variable
2019-03-22 08:57:12.324 UTC [main] SetOrdererEnv -> WARN 002 CORE_LOGGING_LEVEL is no longer supported
, please use the FABRIC_LOGGING_SPEC environment variable
240
root@8592218806e3:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

좌측 바둑판 시계방향(CA – Orderer – peer0 – couchDB0) / 우측 가로배열 아래로(couchDB1 – peer1 – cli)
peer끼리는 계속 싱크를 맞추기 위해 통신을 하기 때문에 log가 계속 올라감
우하단 cli창에 명령어 입력할 경우(invoke, query 등) 각각의 couchDB도 반응(인터넷 접속 가능)