

< Azure 클라우드에서 Blockchain Network 구성하기 >

1. vm 복사하기(window PowerShell 사용)

- 참고사이트 : <https://docs.microsoft.com/ko-kr/azure/virtual-machines/linux/copy-vm#create-a-vm>

1. 내 계정 로그인	az login
2. 원본 VM 중지	az vm deallocate --resource-group KBCN_dheo --name peer1
3. 내 RG에 속한 VM 및 Disk 확인	az vm list -g KBCN_dheo --query '[].(Name:name,DiskName:storageProfile.osDisk.name)' --output table <pre> PS C:\WINDOWS\system32> az vm list -g KBCN_dheo --query '[].(Name:name,DiskName:storageProfile.osDisk.name)' --output table Name DiskName ----- peer1 peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557 peer2 peer2_OsDisk peer3 peer3_OsDisk </pre>
4. 원본 VM의 Disk 복사	az disk create --resource-group KBCN_dheo --name peer2_OsDisk --source peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557 <pre> PS C:\WINDOWS\system32> az disk create --resource-group KBCN_dheo --name peer2_OsDisk --source peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557 { "creationData": { "createOption": "Copy", "imageReference": null, "sourceResourceId": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Compute/disks/peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557", "sourceUri": null, "storageAccountId": null }, "diskIopsReadWrite": 120, "diskMbpsReadWrite": 25, "diskSizeGb": 30, "diskState": "Unattached", "encryptionSettingsCollection": null, "hyperVGeneration": "V1", "id": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Compute/disks/peer2_OsDisk", "location": "koreacentral", "managedBy": null, "name": "peer2_OsDisk", "osType": "Linux", "provisioningState": "Succeeded", "resourceGroup": "KBCN_dheo", "sku": { "name": "Premium_LRS", "tier": "Premium" }, "tags": {}, "timeCreated": "2018-07-10T08:41:22.519449+00:00", "type": "Microsoft.Compute/disks", "zones": null } </pre>
5. Disk 추가 된 것 확인	az disk list --resource-group KBCN_dheo --output table <pre> PS C:\WINDOWS\system32> az disk list --resource-group KBCN_dheo --output table Name ResourceGroup Location Zones Sku OsType SizeGb ProvisioningState ----- peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557 KBCN_dheo koreacentral Premium_LRS Linux 30 Succeeded peer2_OsDisk KBCN_dheo koreacentral Premium_LRS Linux 30 Succeeded peer3_OsDisk KBCN_dheo koreacentral Premium_LRS Linux 30 Succeeded peer4_OsDisk KBCN_dheo koreacentral Premium_LRS Linux 30 Succeeded </pre>
6. VNET 생성하기(건너뛰어도 됨)	az network vnet create --resource-group KBCN_dheo --location koreacentral --name org2-vnet2 --address-prefix 10.2.0.0/16 --subnet-name peer4-subnet4 --subnet-prefix 10.2.4.0/24 <pre> PS C:\WINDOWS\system32> az network vnet create --resource-group KBCN_dheo --location koreacentral --name org2-vnet2 --address-prefix 10.2.0.0/16 --subnet-name peer4-subnet4 --subnet-prefix 10.2.4.0/24 { "newVnet": { "addressSpace": { "addressPrefixes": ["10.2.0.0/16"] }, "dhcpProtection": null, "dnsOptions": { "dnsServers": [] }, "enableDdosProtection": false, "enableVmProtection": false, "etag": "W\"/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2/10.2.0.0/16-10.2.4.0/24\"", "id": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2", "location": "koreacentral", "name": "org2-vnet2", "provisioningState": "Succeeded", "resourceGroup": "KBCN_dheo", "resourceId": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2", "subnets": [{ "addressPrefix": "10.2.4.0/24", "addressPrefixes": null, "delegations": [], "etag": "W\"/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2/subnets/peer4-subnet4\"", "id": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2/subnets/peer4-subnet4", "ipConfigurations": null, "name": "peer4-subnet4", "netGateway": null, "networkSecurityGroup": null, "privateEndpoints": null, "provisioningState": "Succeeded", "resource": null, "resourceGroup": "KBCN_dheo", "resourceNavigationLinks": null, "routeTable": null, "serviceAssociationLinks": null, "serviceEndpointPolicies": null, "serviceEndpoints": null, "type": "Microsoft.Network/virtualNetworks/subnets" }] }, "tags": {}, "type": "Microsoft.Network/virtualNetworks", "virtualNetworkPeering": [] } </pre>

<p>7. PublicIP 생성하기(건너뛰어도 됨) => dns-name은 소문자로 고유하게 작성해야 함</p>	<pre>az network public-ip create --resource-group KBCN_dheo --location koreacentral --name peer4-pip4 --dns-name publicdnsppeer4 --allocation-method static --idle-timeout 4</pre> <pre>PS C:\WINDOWS\system32> az network public-ip create --resource-group KBCN_dheo --location koreacentral --name peer4-pip4 --dns-name publicdnsppeer4 --allocation-method static --idle-timeout 4</pre> <pre>{ "publicIp": { "ddosSettings": null, "dnsSettings": { "domainNameLabel": "publicdnsppeer4", "fqdn": "publicdnsppeer4.koreacentral.cloudapp.azure.com", "reverseFqdn": null }, "etag": "W/\"c353d6bb-7b48-4656-846b-1937184710e1W\"", "id": "/subscriptions/5f8164de-d273-400a-ba7b-4f4d2e62bbe1/resourceGroups/KBCN_dheo/providers/Microsoft.Network/publicIPAddresses/peer4-pip4", "idleTimeoutInMinutes": 4, "ipAddress": "52.141.2.14", "ipConfiguration": null, "ipTags": [], "location": "koreacentral", "name": "peer4-pip4", "provisioningState": "Succeeded", "publicIpAddressVersion": "IPv4", "publicIpAllocationMethod": "Static", "publicIpPrefix": null, "resourceGroup": "KBCN_dheo", "resourceGuid": "cdf1f788-0433-4d09-b2cb-08d69d6ee31a", "sku": { "name": "Basic", "tier": "Regional" }, "tags": null, "type": "Microsoft.Network/publicIPAddresses", "zones": null } }</pre>
<p>8. 서브넷과 연결된 NIC 만들기(건너뛰어도 됨)</p>	<pre>az network nic create --resource-group KBCN_dheo --location koreacentral --name peer4-nic4 --vnet-name org2-vnet2 --subnet peer4-subnet4 --public-ip-address peer4-pip4</pre>
<p>9. VM 생성해 복사된 Disk와 연결</p>	<pre>az vm create --resource-group KBCN_dheo --name peer4 --nics peer4-nic4 --size Standard_D2s_v3 --os-type Linux --attach-os-disk peer4_OsDisk</pre> <pre>PS C:\WINDOWS\system32> az vm create --resource-group KBCN_dheo --name peer4 --nics peer4-nic4 --size Standard_D2s_v3 --os-type Linux --attach-os-disk peer4_OsDisk</pre> <pre>{ "fqdn": "publicdnsppeer4.koreacentral.cloudapp.azure.com", "id": "/subscriptions/5f8164de-d273-400a-ba7b-4f4d2e62bbe1/resourceGroups/KBCN_dheo/providers/Microsoft.Compute/virtualMachines/peer4", "location": "koreacentral", "macAddress": "00-22-48-05-60-73", "powerState": "VM running", "privateIpAddress": "10.2.4.4", "publicIpAddress": "52.141.2.14", "resourceGroup": "KBCN_dheo", "zones": "" }</pre>

2. 생성한 vm들 간 ping 설정하기

- 참고사이트 : <https://docs.microsoft.com/ko-kr/azure/virtual-network/tutorial-connect-virtual-networks-portal>
- ping을 날리는 데 사용되는 프로토콜인 ICMP(Internet Control Message Protocol) 설정 변경
- 기본적으로 icmp 패킷에 대한 응답을 전부 무시하는 1(yes)로 설정되어 있음.

\$ sudo sysctl -w net.ipv4.icmp_echo_ignore_all=0

위 명령어를 입력해주면 0(no)로 바뀌어 ping에 대해 응답하게 됨.

윈도우 powershell에서는 아래와 같이 설정해줄 것!

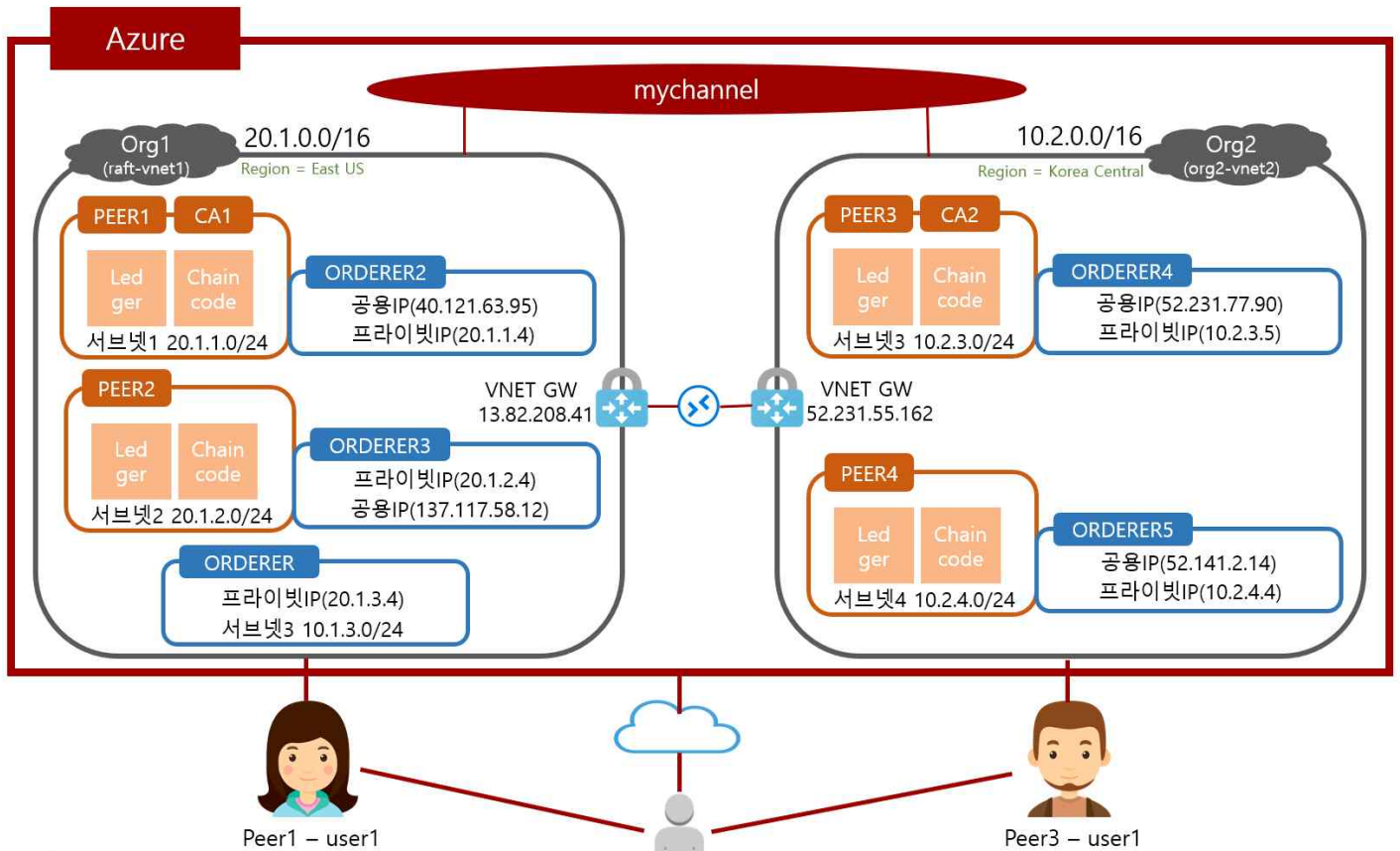
> New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4

- Azure에서 두 가지 유형의 IP 주소를 사용 할 수 있음.
(<https://docs.microsoft.com/ko-kr/azure/virtual-network/virtual-network-ip-addresses-overview-arm>)

공용 IP 주소	Azure 공용 서비스를 포함하여 인터넷과의 통신에 사용.
사설 IP 주소	VPN 게이트웨이 또는 ExpressRoute 회로를 사용하여 Azure로 네트워크를 확장할 때 Azure VNet(가상네트워크) 및 온-프레미스 네트워크 내에서 통신하는데 사용.

3. docker swarm을 활용해 멀티노드 구성하기

(<https://discourse.skcript.com/t/setting-up-a-blockchain-business-network-with-hyperledger-fabric-composer-running-in-multiple-physical-machine/602> 참고 사이트)



네트워크 생성 작업은 Cloud에서 VNET 및 VM을 만들어 연결 후 hyperledger fabric 1.4.2 설치 및 다음 단계 진행

1-1. VM peering 작업	https://docs.microsoft.com/ko-kr/azure/virtual-network/virtual-network-peering-overview 같은 리소스 그룹 및 같은 Region에 있을 때 실시
1-2. VNET to VNET 작업	https://docs.microsoft.com/ko-kr/azure/vpn-gateway/vpn-gateway-how-to-vnet-vnet-resource-manager-portal#vnet-to-vnet 다른 리소스 그룹 또는 다른 Region에 있을 때 실시(같아도 무관) VM의 게이트웨이가 아닌 VNET의 게이트웨이를 설정함으로써 다른 리소스그룹, Region과 통신이 가능(반드시 연결 해줘야 함)

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

* <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)

```
kim@peer1:~/kismi_blockchain$ docker swarm init --advertise-addr 192.168.56.101
Swarm initialized: current node (u0oxdv4gz846kw5wlekjxx70z) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3bhqk7in4empnjdho9l5odpc0x24hzwmc5jh13gfmf94u03k-0y9l8f24z8s0w8vfd0hsip9n9 192.168.56.101:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

\$ docker swarm join-token manager → 매니저(관리자)로 join(하면 아래 복잡한 코드가 뜸)하고자 하는 VM에는 아래 뜨는 코드 입력

② PC2 = 192.168.56.102

\$ docker swarm join --token SWMTKN-1-1zqjcs1f63m3um8q9xr53qbbcczz9b6byjov4l9qrykconvp9z-0y4cuxlgmc56qg6bbcmli8k6g 192.168.56.101:2377

→ 위에 init 했을 때 나오는 코드 그대로 복사해서 붙여넣기(joined 확인) worker로 등록하고자 하는 VM에 입력 할 것!

```
project-b@projectb-VirtualBox:~$ docker swarm join --token SWMTKN-1-1zqjcs1f63m3um8q9xr53qbbcczz9b6byjov4l9qrykconvp9z-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 kismi_bdms → 네트워크 생성 완료 \$docker network create --attachable --driver overlay --subnet=40.121.0.0/24 kismi_bdms

⑤ 생성된 네트워크 및 swarm node 확인하기

\$ docker node ls

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
7ekunth6t5zj5cim85hj6hnc	ordererEastUS	Ready	Active	Reachable	19.03.1
l87h0qvczz0ff6lwo1q0p9jhc *	peer1EastUS	Ready	Active	Leader	19.03.1
x38713nfi0gb03w9jv7r38b88	peer2EastUS	Ready	Active	Reachable	19.03.1
xor0yk1kuc934ufu5ftdmzn2p	peer3	Ready	Active	Reachable	18.09.7
ku2kwaso31u7u1909tbeaqyp	peer4	Ready	Active	Reachable	18.09.7

\$ docker network ls

NETWORK ID	NAME	DRIVER	SCOPE
9c0196b12a28	bridge	bridge	local
2631ffb9fe4a	docker_gwbridge	bridge	local
b60446f591c6	host	host	local
ab573kz8biu6	ingress	overlay	swarm
veg85dt9ga42	kismi_bdms	overlay	swarm
e35100d2ed88	none	null	local

2. docker swarm 만들기

3. VM 네트워크 보안규칙 설정

해당 VM의 역할에 따라 port를 열어줘야 통신 가능

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

4. MSP 생성 및 배포 (orderer, peer 키 생성)	<code>cryptogen generate --config=./crypto-config.yaml</code>
5. genesisblock 생성 (crypto-config 디렉토리와, configtx.yaml 파일이 있는 곳에서 진행) => -profile 옵션은 변경 가능	<code>configtxgen -profile SampleMultiNodeEtcdRaft -outputBlock genesis.block --channelID mychannel</code>
6. channel 설정 (채널 구축을 위한 트랜잭션 생성) => genesis.block 생성한 동일 위치에서 진행	<code>configtxgen -profile TwoOrgsChannel -outputCreateChannelTx mychannel.tx -channelID mychannel</code>
7. 각 조직별 Anchor peer 설정하는 트랜잭션 생성 => 다른 조직 간의 통신을 위해 필요	<code>configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate Org1MSPanchors.tx -channelID mychannel -asOrg Org1MSP</code> <code>configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate Org2MSPanchors.tx -channelID mychannel -asOrg Org2MSP</code>
8. 생성된 MSP 및 초기 파일들 배포 => 모든 노드로 (필요 디렉토리만 보내도 되지만 편의상 전체 dir 복사하기)	<code>mkdir ./channel-artifacts</code> <code>mv genesis.block mychannel.tx Org1MSPanchors.tx Org2MSPanchors.tx ./channel-artifacts/</code> <code>scp -r crypto-config channel-artifacts eodahee@52.141.1.79:~/kismi_blockchain/</code>

9. CA 서버 구동

[illegible]

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

```
docker run -d --rm -it --network="kismi_bdms" --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.ge
nesis.block -e ORDERER_GENERAL_LOCALMSPID=OrdererMSP -e
ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp -e
ORDERER_GENERAL_TLS_ENABLED=true -e
ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/serv
er.key -e
ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/ser
ver.crt -e
ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt
] -e
ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE=/var/hyperledger/or
derer/tls/server.crt -e
ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY=/var/hyperledger/or
derer/tls/server.key -e
ORDERER_GENERAL_CLUSTER_ROOTCAS=[/var/hyperledger/orderer/tls
/ca.crt] -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -v
$(pwd)/channel-artifacts/genesis.block:/var/hyperledger/orderer/ordere
r.genesis.block -v
$(pwd)/crypto-config/ordererOrganizations/example.com/orderers/ord
erer.example.com/msp:/var/hyperledger/orderer/msp -v
$(pwd)/crypto-config/ordererOrganizations/example.com/orderers/ord
erer.example.com/tls:/var/hyperledger/orderer/tls -w
/opt/gopath/src/github.com/hyperledger/fabric
hyperledger/fabric-orderer:1.4.2 orderer .
```

```
Consensus = map[MD5]:[var/hyperledger/production/orderer/etcdraft/v0. Snapshot:/var/hyperledger/production/orderer/etcdraft/snapshot]
Operations.ListenAddress = "0.0.0.0:7050"
Operations.TLS.Enabled = false
Operations.TLS.PrivateKey = ""
Operations.TLS.Certificate = ""
Operations.TLS.RootCAs = []
Operations.TLS.ClientRootCAs = []
Operations.TLS.ClientRootCAs = []
Metrics.Provider = "disabled"
Metrics.Statsd.Address = "udp"
Metrics.Statsd.Address = "127.0.0.1:8125"
Metrics.Statsd.WritelnInterval = 30s
Metrics.Statsd.Prefix = ""
2019-07-15 08:03:46.206 UTC [orderer.common.server] InitializeServerConfig -> INFO 003 Starting orderer with TLS enabled
2019-07-15 08:03:46.199 UTC [fabricstorage] seedblockfilemgr -> INFO 004 Getting block information from block storage
2019-07-15 08:03:46.205 UTC [orderer.common.multichannel] Initialize -> INFO 005 Starting system channel 'testchainid' with genesis block hash ac476ba32576e65dc4ada0a46de4cf6a336e9802b191d69979760d2
and orderer type solo
2019-07-15 08:03:46.205 UTC [orderer.common.server] Start -> INFO 006 Starting orderer:
Version: 1.4.1
Commit SHA: 97074a7
Go version: go1.11.5
OS/Arch: Linux/amd64
2019-07-15 08:03:46.205 UTC [orderer.common.server] Start -> INFO 007 Beginning to serve requests
2019-07-15 08:03:46.630 UTC [comm.grpc.server] 1 -> INFO 008 streaming call completed grpc.serviceorderer.AtomicBroadcast grpc.methodBroadcast grpc.peer_address=10.0.0.70:46544 grpc.code=OK grpc.call_dura
tione24.487646ms
2019-07-15 08:03:46.642 UTC [fabricstorage] seedblockfilemgr -> INFO 009 Getting block information from block storage
2019-07-15 08:03:46.677 UTC [orderer.common.multichannel] newchain -> INFO 00a Created and starting new chain mychannel
2019-07-15 08:03:46.680 UTC [comm.grpc.server] 1 -> INFO 00b streaming call completed grpc.serviceorderer.AtomicBroadcast grpc.methodDeliver grpc.peer_address=10.0.0.70:46542 grpc.code=OK grpc.call_dura
tion272.591439ms
2019-07-15 08:03:41.945 UTC [orderer.common.broadcast] Handle -> WARN 00c Error reading from 10.0.0.70:46570: rpc error: code = Canceled desc = context canceled
2019-07-15 08:03:41.945 UTC [comm.grpc.server] 1 -> INFO 00d streaming call completed grpc.serviceorderer.AtomicBroadcast grpc.methodBroadcast grpc.peer_address=10.0.0.70:46570 error=rpc error: code = Canceled desc = context canceled grpc.code=Canceled grpc.call_duration=8.771310ms
2019-07-15 08:03:41.945 UTC [comm.grpc.server] 1 -> INFO 00e streaming call completed grpc.serviceorderer.AtomicBroadcast grpc.methodDeliver grpc.peer_address=10.0.0.70:46568 error=rpc error: code = Canceled desc = context canceled grpc.code=Canceled grpc.call_duration=13.24222ms
2019-07-15 08:03:42.020 UTC [comm.grpc.server] 1 -> INFO 010 Error reading from 10.0.0.70:46572: rpc error: code = Canceled desc = context canceled
2019-07-15 08:03:42.020 UTC [comm.grpc.server] 1 -> INFO 011 streaming call completed grpc.serviceorderer.AtomicBroadcast grpc.methodDeliver grpc.peer_address=10.0.0.70:46572 error=rpc error: code = Canceled desc = context canceled grpc.code=Canceled grpc.call_duration=13.92272ms
```

10. Orderer 구동

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

```
docker run -d --rm -it --network="kismi_bdms" --name couchdb1
-p 5984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms
hyperledger/fabric-couchdb
```

```
docker run -d --rm -it --link
orderer.example.com:orderer.example.com --network="kismi_bdms"
--name peer0.org1.example.com -p 8051:7051 -p 8053:7053 -e
CORE_LEDGER_STATE_STATEDATABASE=CouchDB -e
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchd
b1:5984 -e CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= -e
CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= -e
CORE_PEER_ADDRESS=peer0.org1.example.com:7051 -e
CORE_PEER_LISTENADDRESS=0.0.0.0:7051 -e
CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7052 -e
CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052 -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:705
0 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:70
51 -e CORE_PEER_TLS_ENABLED=true -e
CORE_PEER_GOSSIP_USELEADERELECTION=true -e
CORE_PEER_GOSSIP_ORGLEADER=false -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt -e
CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key -e
CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt -e
CORE_PEER_LOCALMSPID=Org1MSP -v /var/run:/host/var/run/ -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer
0.org1.example.com/msp:/etc/hyperledger/fabric/msp -v
$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer
0.org1.example.com/tls:/etc/hyperledger/fabric/tls -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
hyperledger/fabric-peer:1.4.2 peer node start
```

11. CouchDB1 및 Peer1 구동

```
2019-07-15 08:33:06.844 UTC [gossip.discovery] learnExistingMembers -> DEBU 5a678 Exiting
2019-07-15 08:33:06.844 UTC [gossip.discovery] handleAliveMessage -> DEBU 5a679 Exiting
2019-07-15 08:33:06.844 UTC [gossip.discovery] handleMsgFromComm -> DEBU 5a67a Exiting
2019-07-15 08:33:06.851 UTC [gossip.comm] func1 -> DEBU 5a67b Got message: GossipMessage: tag:EMPTY a
live msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"3\t\241\273\300P2\372\3521\
246EM\212\9\363D!\<022\267\273A$\2631\301\233\231" > timestamp:<inc_num:1563177751283340186 seq_num:8
36 > > , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.851 UTC [gossip.gossip] handleMessage -> DEBU 5a67c Entering, 10.0.0.74:7051 366e
3ea2912ddcab7312d73eb6cde6fd01a19675a591486dcad8e112954 sent us GossipMessage: tag:EMPTY alive m
sg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"3\t\241\273\300P2\372\3521\246EM\
212\9\363D!\<022\267\273A$\2631\301\233\231" > timestamp:<inc_num:1563177751283340186 seq_num:836 > >
, Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.851 UTC [gossip.gossip] handleMessage -> DEBU 5a67d Exiting
2019-07-15 08:33:06.851 UTC [gossip.discovery] handleMsgFromComm -> DEBU 5a67e Got message: GossipMes
sage: tag:EMPTY alive msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"3\t\241\273\300P2\372\3521\246EM\
212\9\363D!\<022\267\273A$\2631\301\233\231" > timestamp:<inc_num:1563177751283340186 seq_n
um:836 > > , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.851 UTC [gossip.discovery] handleMsgFromComm -> DEBU 5a67f Exiting
2019-07-15 08:33:06.854 UTC [msp] DeserializeIdentity -> DEBU 5a680 Obtaining identity
2019-07-15 08:33:06.854 UTC [msp] DeserializeIdentity -> DEBU 5a681 Obtaining identity
2019-07-15 08:33:06.854 UTC [msp] DeserializeIdentity -> DEBU 5a682 Obtaining identity
2019-07-15 08:33:06.854 UTC [gossip.comm] Send -> DEBU 5a683 Entering, sending GossipMessage: tag:EMP
TY alive msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"3\t\241\273\300P2\372\3521\246EM\
212\9\363D!\<022\267\273A$\2631\301\233\231" > timestamp:<inc_num:1563177751283340186 seq_n
um:836 > > , Envelope: 84 bytes, Signature: 71 bytes to 1 peers
2019-07-15 08:33:06.854 UTC [msp] DeserializeIdentity -> DEBU 5a684 Obtaining identity
2019-07-15 08:33:06.854 UTC [gossip.comm] Send -> DEBU 5a685 Entering, sending GossipMessage: tag:EMP
TY alive msg:<membership:<endpoint:"peer1.org1.example.com:7051" pki_id:"3\t\241\273\300P2\372\3521\246EM\
212\9\363D!\<022\267\273A$\2631\301\233\231" > timestamp:<inc_num:1563177751283340186 seq_n
um:836 > > , Envelope: 84 bytes, Signature: 71 bytes to 1 peers
```


3. docker swarm을 활용해 멀티노드 구성하기(이어서)

12. CouchDB2 및 Peer2 구동

```
docker run -d --rm -it --network="kismi_bdms" --name couchdb2
-p 6984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms
hyperledger/fabric-couchdb

docker run -d --rm -it --link
orderer.example.com:orderer.example.com ₩
--network="kismi_bdms" --name peer1.org1.example.com -p
9051:7051 -p 9053:7053₩
-e CORE_LEDGER_STATE_STATEDATABASE=CouchDB ₩
-e
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchd
b2:5984 ₩
-e CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= ₩
-e CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= ₩
-e CORE_PEER_ADDRESS=peer1.org1.example.com:7051 ₩
-e CORE_PEER_LISTENADDRESS=0.0.0.0:7051 ₩
-e CORE_PEER_CHAINCODEADDRESS=peer1.org1.example.com:7052
₩
-e CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052 ₩
-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:705
0 ₩
-e CORE_PEER_GOSSIP_IGNORESECURITY=true ₩
-e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms
₩
-e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:70
51 ₩
-e CORE_PEER_TLS_ENABLED=true -e
CORE_PEER_GOSSIP_USELEADERELECTION=true ₩
-e CORE_PEER_GOSSIP_ORGLEADER=false -e
CORE_PEER_PROFILE_ENABLED=true ₩
-e CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt ₩
-e CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key ₩
-e CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
₩
-e CORE_PEER_LOCALMSPID=Org1MSP -v /var/run:/host/var/run/ ₩
-v
"$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/pee
r1.org1.example.com/msp":/etc/hyperledger/fabric/msp ₩
-v
"$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/pee
r1.org1.example.com/tls":/etc/hyperledger/fabric/tls ₩
-w /opt/gopath/src/github.com/hyperledger/fabric/peer ₩
hyperledger/fabric-peer:1.4.2 peer node start
```

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

13. CouchDB3 및 Peer3 구동

```
docker run -d --rm -it --network="kismi_bdms" --name couchdb3
-p 5984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms
hyperledger/fabric-couchdb

docker run -d --rm -it --link
orderer.example.com:orderer.example.com --network="kismi_bdms"
--name peer0.org2.example.com -p 8051:7051 -p 8053:7053 -e
CORE_LEDGER_STATE_STATEDATABASE=CouchDB -e
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchd
b3:5984 -e CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= -e
CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= -e
CORE_PEER_ADDRESS=peer0.org2.example.com:7051 -e
CORE_PEER_LISTENADDRESS=0.0.0.0:7051 -e
CORE_PEER_CHAINCODEADDRESS=peer0.org2.example.com:7052 -e
CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052 -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e
FABRIC_LOGGING_SPEC=DEBUG -e
CORE_PEER_NETWORKID=peer0.org2.example.com -e
CORE_NEXT=true -e CORE_PEER_ENDORSER_ENABLED=true -e
CORE_PEER_ID=peer0.org2.example.com -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:705
0 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org2.example.com:70
51 -e CORE_PEER_TLS_ENABLED=true -e
CORE_PEER_GOSSIP_USELEADERELECTION=true -e
CORE_PEER_GOSSIP_ORGLEADER=false -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt -e
CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key -e
CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt -e
CORE_PEER_LOCALMSPID=Org2MSP -v /var/run:/host/var/run/ -v
$(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer
0.org2.example.com/msp:/etc/hyperledger/fabric/msp -v
$(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer
0.org2.example.com/tls:/etc/hyperledger/fabric/tls -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
hyperledger/fabric-peer:1.4.2 peer node start
```

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

```
docker run -d --rm -it --network="kismi_bdms" --name couchdb4
-p 6984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms
hyperledger/fabric-couchdb
```

```
docker run -d --rm -it --link
orderer.example.com:orderer.example.com --network="kismi_bdms"
--name peer1.org2.example.com -p 9051:7051 -p 9053:7053 -e
CORE_LEDGER_STATE_STATEDATABASE=CouchDB -e
CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchd
b4:5984 -e CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME= -e
CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD= -e
CORE_PEER_ADDRESS=peer1.org2.example.com:7051 -e
CORE_PEER_LISTENADDRESS=0.0.0.0:7051 -e
CORE_PEER_CHAINCODEADDRESS=peer1.org2.example.com:7052 -e
CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052 -e
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock -e
FABRIC_LOGGING_SPEC=DEBUG -e
CORE_PEER_NETWORKID=peer1.org2.example.com -e
CORE_NEXT=true -e CORE_PEER_ENDORSER_ENABLED=true -e
CORE_PEER_ID=peer0.org2.example.com -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_COMMITTER_LEDGER_ORDERER=orderer.example.com:705
0 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org2.example.com:70
51 -e CORE_PEER_TLS_ENABLED=true -e
CORE_PEER_GOSSIP_USELEADERELECTION=true -e
CORE_PEER_GOSSIP_ORGLEADER=false -e
CORE_PEER_PROFILE_ENABLED=true -e
CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt -e
CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key -e
CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt -e
CORE_PEER_LOCALMSPID=Org2MSP -v /var/run:/host/var/run/ -v
$(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer
1.org2.example.com/msp:/etc/hyperledger/fabric/msp -v
$(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer
1.org2.example.com/tls:/etc/hyperledger/fabric/tls -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
hyperledger/fabric-peer:1.4.2 peer node start
```

```
2019-07-15 08:33:06.852 UTC [gossip.gossip] handleMessage -> DEBU 459e1 Exiting
2019-07-15 08:33:06.852 UTC [gossip.discovery] handleMessageFromComm -> DEBU 459e2 Got message: GossipMes
sage: tag:EMPTY alive msg:membership: endpoint: "peer1.org1.example.com:7051" pki id: "3\t\t241\t273\t327
\t300P2\t372\t\3521\t246EM\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231" > timestamp: <inc num:156317775128
3340186 seq num:836 > > , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.852 UTC [gossip.discovery] handleMessageFromComm -> DEBU 459e3 Exiting
2019-07-15 08:33:06.855 UTC [gossip.gossip] handleMessage -> DEBU 459e4 Entering, 10.0.0.68:52248 40a
842f1f6c752e5ac19b9b50647f47b7052d6d03261a84b7b22dbefc0576fb1 sent us GossipMessage: tag:EMPTY alive
msg:membership: endpoint: "peer1.org1.example.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM
\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231" > timestamp: <inc num:1563177751283340186 seq num:836 >
> , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.855 UTC [gossip.gossip] handleMessage -> DEBU 459e5 Exiting
2019-07-15 08:33:06.855 UTC [gossip.discovery] handleMessageFromComm -> DEBU 459e6 Got message: GossipMes
sage: tag:EMPTY alive msg:membership: endpoint: "peer1.org1.example.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM
\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231" > timestamp: <inc num:1563177751283340186 seq num:836 >
> , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.855 UTC [gossip.discovery] handleMessageFromComm -> DEBU 459e7 Exiting
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459e8 Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459e9 Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459ea Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459eb Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459ec Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459ed Obtaining identity
2019-07-15 08:33:06.856 UTC [gossip.comm] Send -> DEBU 459ee Entering, sending GossipMessage: tag:EMP
TY alive msg:membership: endpoint: "peer1.org1.example.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM
\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231" > timestamp: <inc num:1563177751283340186 seq n
um:836 > > , Envelope: 84 bytes, Signature: 71 bytes to 1 peers
2019-07-15 08:33:06.856 UTC [gossip.comm] sendToEndpoint -> DEBU 459ef Obtaining identity
2019-07-15 08:33:06.856 UTC [msp] DeserializeIdentity -> DEBU 459f0 Obtaining identity
2019-07-15 08:33:06.856 UTC [gossip.comm] Send -> DEBU 459f1 Entering, sending GossipMessage: tag:EMP
TY alive msg:membership: endpoint: "peer1.org1.example.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM
\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231" > timestamp: <inc num:1563177751283340186 seq n
um:836 > > , Envelope: 84 bytes, Signature: 71 bytes to 1 peers
2019-07-15 08:33:06.856 UTC [gossip.comm] sendToEndpoint -> DEBU 459f2 Entering, Sending to peer0.org
2.example.com:7051 , msg: GossipMessage: tag:EMPTY alive msg:membership: endpoint: "peer1.org1.examp
le.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231
" > timestamp: <inc num:1563177751283340186 seq num:836 > > , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.856 UTC [gossip.comm] sendToEndpoint -> DEBU 459f3 Exiting
2019-07-15 08:33:06.856 UTC [gossip.comm] sendToEndpoint -> DEBU 459f4 Entering, Sending to peer0.org
2.example.com:7051 , msg: GossipMessage: tag:EMPTY alive msg:membership: endpoint: "peer1.org1.examp
le.com:7051" pki id: "3\t\t241\t273\t327\t300P2\t372\t\3521\t246EM\t212\t9\t363D\t\t\022\t267\t273AS\t2631\t301\t233\t231
" > timestamp: <inc num:1563177751283340186 seq num:836 > > , Envelope: 84 bytes, Signature: 71 bytes
2019-07-15 08:33:06.856 UTC [gossip.comm] sendToEndpoint -> DEBU 459f5 Exiting
```

14. CouchDB4 및 Peer4 구동

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

```
docker run --rm -it --network="kismi_bdms" --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=true -e
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.or
g1.example.com/tls/server.crt -e
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fab
ric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.
example.com/tls/server.key -e
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledg
er/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/pee
r0.org1.example.com/tls/ca.crt -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/hyperledg
er/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admi
n@org1.example.com/msp -e
CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -v
/var/run:/host/var/run/ -v
$(pwd)/chaincode:/opt/gopath/src/github.com/chaincode -v
$(pwd)/crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/p
eer/crypto/ -v
$(pwd)/scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scri
pts/ -v
$(pwd)/channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabri
c/peer/channel-artifacts -w
/opt/gopath/src/github.com/hyperledger/fabric/peer
hyperledger/fabric-tools:1.4.2 /bin/bash -c './scripts/script.sh'
```

[illegible]

15. CLI 구동 및 channel
create, join, chaincode
install, instantiated 등 작업
(완료되면 맨 뒤에 -c
이후만 지우고 /bin/bash까지
실행하면 di 컨테이너로 입성)

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

16. admin, user 등록을
통해 sdk를 활용한 통신

```

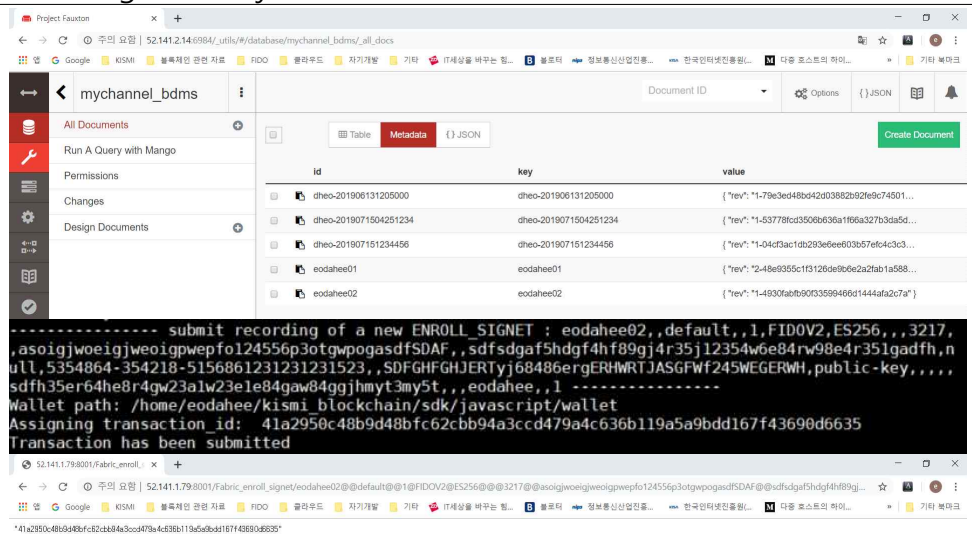
2019/07/15 08:27:42 [DEBUG] Registering user id: user1
2019/07/15 08:27:42 [DEBUG] Max enrollment value verification - User specified max enrollment: 1, CA
max enrollment: -1
2019/07/15 08:27:42 [DEBUG] DB: Getting identity user1
2019/07/15 08:27:42 [DEBUG] DB: Add identity user1
2019/07/15 08:27:42 [DEBUG] Successfully added identity user1 to the database
2019/07/15 08:27:42 [INFO] 172.18.0.1:34394 POST /api/v1/register 201 0 "OK"
2019/07/15 08:27:42 [DEBUG] Received request for /api/v1/enroll
2019/07/15 08:27:42 [DEBUG] ca.Config: &{Version:1.4.1 Cfg:{Identities:{PasswordAttempts:10 AllowRemo
ve:false} Affiliations:{AllowRemove:false}} CA:{Name:(CAname.example.com Keyfile:/etc/hyperledger/fabric-ca-
server/server.keys Certfile:/etc/hyperledger/fabric-ca-server/cert.pem Chainfile:/etc/hyperledger/fabric-ca-se
rver/ca-chain.pem) Signing:oc090307960 CSR:{CN:cn.org.example.com Names:[{:C:US ST:North Carolina L:
O:Hyperledger OU:Fabric SerialNumber:}] Hosts:[es0945166bb5f localhost]} KeyRequest:0xc0002b97a0 CA:8x
c0002b9520 SerialNumber:} Registry:{MaxEnrollments:-1 Identities:[{ Name:**** Pass:**** Type:Client A
ffiliation: MaxEnrollments:0 Attrs:{map[hf.Registrar.Roles:* hf.Registrar.DelegateRoles:* hf.Revoker:1
hf.IntermediateCA:1 hf.GenCRIT:1 hf.Registrar.Attributes:* hf.AttributionMgr:1]}]} Affiliations:{map
[depmont1 depmont2] Order:[depmont1]} LDAP:{ Enabled:false URL:ldap://****:****@host-s
upport/-base?UserFilter=(uid=%s) GroupFilter:(memberUid=%s) Attribute:[{uid member}] } } map[groups
:[{]] TLS:{false []}} DB:{ Type:sqLite3 DataSource:/etc/hyperledger/fabric-ca-server/fabric-ca-
server.db TLS:{false []}} CSP:oc0002b8360 Client:<nill> Intermediate:{ParentServer:{ URL:CA
Name:} TLS:{Enabled:false Certfiles:[]} Client:{Keyfile: Certfile:}} Enrollment:{ Name: Secret:****
CAName: AttrReqs:[] Profile: Label: CSR-<nill> Type:x509 }} CRL:{Expiry:24h0m0s} Idemix:{IssuerPublic
Key:/etc/hyperledger/fabric-ca-server/IssuerPublicKey IssuerSecretKeyfile:/etc/hyperledger/fabric-
ca-server/msp/keystore/IssuerSecretKey RevocationPublicKeyfile:/etc/hyperledger/fabric-ca-server/Iss
uerRevocationPublicKey RevocationPrivateKeyfile:/etc/hyperledger/fabric-ca-server/msp/keystore/Issuer
RevocationPrivateKey RHPoolSize:1000 NonceExpiration:15s NoncesSweepInterval:15m}}
2019/07/15 08:27:42 [DEBUG] DB: Getting identity user1
2019/07/15 08:27:42 [DEBUG] DB: Login user user1 with max enrollments of 1 and state of 0
2019/07/15 08:27:42 [DEBUG] DB: Identity user1 successfully logged in
2019/07/15 08:27:42 [DEBUG] DB: Get identity user1
2019/07/15 08:27:42 [DEBUG] Processing sign request: id=user1, CommonName=user1, Subject=<nill>
2019/07/15 08:27:42 [DEBUG] Request is not for a CA signing certificate
2019/07/15 08:27:42 [DEBUG] Checking CSR fields to make sure that they do not exceed maximum characte
r limits
2019/07/15 08:27:42 [DEBUG] Finished processing sign request
2019/07/15 08:27:42 [DEBUG] DB: Getting identity user1
2019/07/15 08:27:42 [DEBUG] Attribute extension being added to certificate is: &{ID:[1 2 3 4 5 6 7 8
1] Critical:false Value:7b261f7472322a7b268662e41666696c966174696fe6223a267267312664657061727
46d56567431222c2268662e456726f6c646d656e744944223a227573657231222c2268662e4574797065723a22636965674
227d7d}
2019/07/15 08:27:42 [DEBUG] Adding attribute extension to CSR: &{ID:[1 2 3 4 5 6 7 8 1] Critical:fals
e Value:7b261f7472322a7b268662e41666696c966174696fe6223a26726731266465706172746d56567431222c2
268662e456726f6c646d656e744944223a227573657231222c2268662e4574797065723a226369656e74727d7d}
2019/07/15 08:27:42 [INFO] signed certificate with serial number 54404712712162825647889543335703226
390509462015
2019/07/15 08:27:42 [DEBUG] DB: Insert Certificate
2019/07/15 08:27:42 [DEBUG] Saved serial number as hex 5f4bacbe1b9153f17b9d21980f45672d53b127
90509462015
2019/07/15 08:27:42 [DEBUG] saved certificate with serial number 54404712712162825647889543335703226
390509462015
2019/07/15 08:27:42 [DEBUG] Successfully incremented state for identity user1 to 1
2019/07/15 08:27:42 [INFO] 172.18.0.1:34492 POST /api/v1/enroll 201 0 "OK"

```

node enrollAdmin.js

```
node registerUser.js
```

17. couchDB 및 web 확인



connection.js와 enrollAdmin.js regisgerUser.js 파일을 잘 맞게 수정
 사용중인 port 죽이기 => \$ fuser -kn tcp 포트번호
 tcp 해당 포트번호를 가진 것을 k(kill)

linux(Ubuntu) VM 방화벽 설정(<https://webdir.tistory.com/206>)

허용 => `sudo ufw allow` 포트번호/방식

차단 => `sudo ufw deny` 포트번호/방식

IP table 확인하기(<https://bluese05.tistory.com/53>)

```
$ sudo iptables -t nat -L -n
```

도커 컨테이너간 통신확인을 위한 ping과 ifconfig 설치

```
# apt-get update && apt-get install -y iputils-ping
```

```
# apt-get update && apt-get install -y net-tools
```

* 최종적으로 generate.sh 파일을 만들어 4~8의 과정을 단순화 하고
9~15번은 각 VM의 역할에 맞게 run peer0 orderer0.sh 파일을 만들어 구동

4. 네트워크 구성 간 발생한 오류들

① 권한 오류 문제

```
2019-07-29 08:43:01.068 UTC [grpc] DialContext -> DEBU 047 parsed scheme: **
2019-07-29 08:43:01.068 UTC [grpc] DialContext -> DEBU 048 scheme ** not registered, fallback to default scheme
2019-07-29 08:43:01.068 UTC [grpc] watcher -> DEBU 049 ccResolverWrapper: sending new addresses to cc: [{orderer.example.com:7050 0 <nil>}]
2019-07-29 08:43:01.068 UTC [grpc] switchBalancer -> DEBU 04a ClientConn switching balancer to "pick_first"
2019-07-29 08:43:01.068 UTC [grpc] HandleSubConnStateChange -> DEBU 04b pickfirstBalancer: HandleSubConnStateChange: 0xc00039af90, CONNECTING
2019-07-29 08:43:01.072 UTC [grpc] HandleSubConnStateChange -> DEBU 04c pickfirstBalancer: HandleSubConnStateChange: 0xc00039af90, READY
Error: got unexpected status: FORBIDDEN -- implicit policy evaluation failed - 0 sub-policies were satisfied, but this policy requires 1 of the 'Writers' sub-policies to be satisfied: permission denied
!!!!!!!!!!!!!! Channel creation failed !!!!!!!!!!!!!!!
===== ERROR !!! FAILED to execute End-2-End Scenario =====
```

[channel: mychannel] Rejecting broadcast of config message from 10.0.0.27:56188

because of error: implicit policy evaluation failed - 0 sub-policies were satisfied, but this policy requires 1 of the 'Writers' sub-policies to be satisfied: permission denied

<https://stackoverflow.com/questions/54716671/failed-to-reach-implicit-threshold-of-1-sub-policies-required-1-remaining-perm>

genesis.block 생성 시 만드는 channel은 내가 생성 할 채널명이 아닌 다른 채널로 실행해야 함. 'byfn-sys-channel'로 만들어 오류 해결!!

mychannel로 생성 시 이미 채널이 만들어지고 동일한 채널 식별을 사용해 channel.tx 파일을 찾아서 오류 발생하는 것(기본적으로 ./byfn down 및 docker volume prune 상태에서 진행) (위 image는 cli에서 발생한 오류 로그 / 밑에 문자는 orderer에 찍힌 오류 로그)

② port forwarding 문제

failed connecting to orderer5.example.com:7050: failed to create new connection: connection error: desc = "transport: error while dialing: dial tcp 10.0.0.42:7050: connect: connection refused"

각 VM에서 docker run 할 때 -p 옵션을 통해 지정해주는 포트가 외부와 도커 컨테이너를 연결해주는 역할. 보통 CA는 7054, peer는 7051과 7053, orderer는 7050임. 포트포워딩 시 8050:7050이라고 입력하면 외부에서 8050 포트로 들어오는 모든 값들은 컨테이너의 7050 포트로 연결해준다는 의미(orderer기준)

그러므로, 초기 MSP 생성 시 쓰이는(genesis block, channel.tx 생성 등) configtx.yaml 파일에서 설정을 잘 해줘야 함!

```
SampleMultiNodeEtcdRaft:
  <<: *ChannelDefaults
  Capabilities:
    <<: *ChannelCapabilities
  Orderer:
    <<: *OrdererDefaults
    OrdererType: etcdraft
    EtcdRaft:
      Consenters:
        - Host: orderer.example.com
          Port: 7050
          ClientTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
          ServerTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
        - Host: orderer2.example.com
          Port: 8050
          ClientTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/tls/server.crt
          ServerTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/tls/server.crt
        - Host: orderer3.example.com
          Port: 9050
          ClientTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer3.example.com/tls/server.crt
          ServerTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer3.example.com/tls/server.crt
        - Host: orderer4.example.com
          Port: 10050
          ClientTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer4.example.com/tls/server.crt
          ServerTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer4.example.com/tls/server.crt
        - Host: orderer5.example.com
          Port: 11050
          ClientTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer5.example.com/tls/server.crt
          ServerTLS-cert: crypto-config/ordererOrganizations/example.com/orderers/orderer5.example.com/tls/server.crt
      Addresses:
        - orderer.example.com:7050
        - orderer2.example.com:8050
        - orderer3.example.com:9050
        - orderer4.example.com:10050
        - orderer5.example.com:11050
    Organizations:
      - *OrdererOrg
    Capabilities:
      <<: *OrdererCapabilities
  Application:
    <<: *ApplicationDefaults
    Organizations:
      - <<: *OrdererOrg
  Consortiums:
    SampleConsortium:
      Organizations:
        - *Org1
        - *Org2
```

혹시라도 이 파일을 잘못 건드리면 channel 생성 간 implicit authn 오류 발생하므로 주의!

③ CA 문제

enrollAdmin.js 및 registerUser.js 실행 할 때 CA가 2개이므로 각각 connection.json 파일을 다르게 작성해줘야 함.

또한 포트포워딩을 7054:7054(CA1)와 8054:7054(CA2)로 지정해줘야 각각 정상작동 가능

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
5db58fe15f86	hyperledger/fabric-ca:1.4.1	"sh -c 'fabric-ca-se..."	About an hour ago	Up About an hour	0.0.0.0:8054->7054/tcp

```
{
  "name": "kismi_bdms",
  "x-type": "h1fv1",
  "version": "1.0.0",
  "client": {
    "organization": "Org1",
    "connection": {
      "timeout": {
        "peer": {
          "endorser": "300"
        },
        "orderer": "300"
      }
    }
  }
}
```

```
{
  "name": "kismi_bdms",
  "x-type": "h1fv1",
  "version": "1.0.0",
  "client": {
    "organization": "Org2",
    "connection": {
      "timeout": {
        "peer": {
          "endorser": "300"
        },
        "orderer": "300"
      }
    }
  }
}
```

CA1에 속한 peer1의 connection.json 클라이언트는 Org1으로,
CA2에 속한 peer3의 connection.json 클라이언트는 Org2로 각각 달리 지정해줘야
권한 문제가 발생하지 않고 등록이 가능!!

enrollAdmin.js와 registerUser.js에서도 각각 ca명(ca.org1.example.com과 ca.org2.example.com)과
X509WalletMixin의 MSP(Org1MSP과 Org2MSP) 수정 필요!!

④ git 업로드 문제

git add 및 git commit -m "~~~" 후 최종적으로 git push 명령어 입력 시 해당
bitbucket에 접근 할 수 없다는 오류 발생(host 못 찾음)

```
eodahee@peer3:~/kismi_blockchain_raft_peer3$ git push
fatal: unable to access 'https://dheo@bitbucket.org/dheo/kismi_blockchain_raft_peer3.git/': Could not resolve host: bitbucket.org
```

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
#nameserver 10.2.0.1
nameserver 8.8.8.8
nameserver 8.8.4.4
search reddog.microsoft.com
```

/etc/resolv.conf 파일에 DNS 매핑 테이블이 존재하지 않아 발생하는 문제
nameserver 8.8.8.8와 nameserver 8.8.4.4를 추가하고 다시 실행해보면 오류 해결
resolv.conf 파일은 요청 할 DNS 서버를 지정 할 때 사용(아래 참고 블로그)

<https://www.nicovideo.jp/watch/N1028696-20808208?tagNo=0&partNo=0&viewDe=8&urlPage=18&pollPage=18&on=polw>

- 8.8.8.8 = Google의 퍼블릭 DNS 서비스(IP주소와 Domain 매핑) 중 IPv4 주소
- 2001:4860:4860::8888 = Google의 퍼블릭 DNS 서비스(IP주소와 Domain 매핑) 중 IPv6 주소

```
docker stop $(docker ps -qa) && docker rm $(docker ps -qa)
rm -rf crypto-config/* channel-artifacts/*
```

- #10.1.1.4 peer1
- #10.1.5.4 orderer
- 20.1.1.4 peer1
- 20.1.2.4 peer2
- 20.1.3.4 orderer
- 10.2.3.5 peer3
- 10.2.4.4 peer4