〈 Azure 클라우드에서 Blockchain Network 구성하기 〉

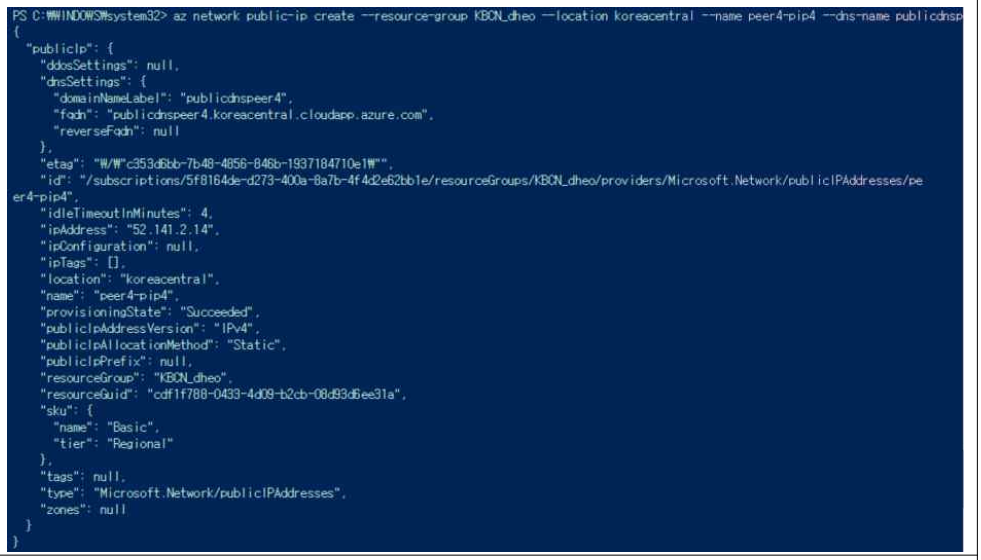
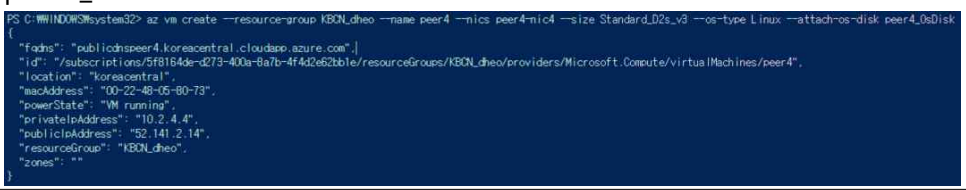
1. vm 복사하기(window PowerShell 사용)
   - 참고사이트 : https://docs.microsoft.com/ko-kr/azure/virtual-machines/linux/copy-vm#create-a-vm

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 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

4. 원본 VM의 Disk 복사	az disk create --resource-group KBCN_dheo --name peer2_OsDisk --source peer1_OsDisk_1_6c6195a0a66045c9bf82fd80986ab557
	PS C:₩WINDOWS₩system32> az disk create --resource-group KBCN_dheo --name peer4_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/peer4_OsDisk",
	"location": "koreacentral",
	"managedBy": null,
	"name": "peer4_OsDisk",
	"osType": "Linux",
	"provisioningState": "Succeeded",
	"resourceGroup": "KBCN_dheo",
	"sku": {
	"name": "Premium_LRS",
	"tier": "Premium"
	},
	"tags": {},
	"timeCreated": "2019-07-10T08:41:22.519449+00:00",
	"type": "Microsoft.Compute/disks",
	"zones": null
	}

5. Disk 추가 된 것 확인	az disk list --resource-group KBCN_dheo --output table 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

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
	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"
	]
	},
	"ddosProtectionPlan": null,
	"dhcpOptions": {
	"dnsServers": []
	},
	"enableDdosProtection": false,
	"enableVmProtection": false,
	"etag": "₩/₩"9c7deab9-7480-4797-836c-ec8dac08346df"",
	"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",
	"resourceGuid": "79210fbb-dd58-41f3-97db-b847aacdfceb",
	"subnets": [
	{
	"addressPrefix": "10.2.4.0/24",
	"addressPrefixes": null,
	"delegations": [],
	"etag": "₩/₩"9c7deab9-7480-4797-836c-ec8dac08346df"",
	"id": "/subscriptions/5f8164de-d273-400a-8a7b-4f4d2e62bb1e/resourceGroups/KBCN_dheo/providers/Microsoft.Network/virtualNetworks/org2-vnet2/subnets/peer4-subnet4",
	"ipConfigurationProfiles": null,
	"ipConfigurations": null,
	"name": "peer4-subnet4",
	"natGateway": null,
	"networkSecurityGroup": null,
	"privateEndpoints": null,
	"provisioningState": "Succeeded",
	"purpose": null,
	"resourceGroup": "KBCN_dheo",
	"resourceNavigationLinks": null,
	"routeTable": null,
	"serviceAssociationLinks": null,
	"serviceEndpointPolicies": null,
	"serviceEndpoints": null,
	"type": "Microsoft.Network/virtualNetworks/subnets"
	}
	],
	"tags": {},
	"type": "Microsoft.Network/virtualNetworks",
	"virtualNetworkPeerings": []
	}
	}

7. PublicIP 생성하기(건너뛰어도 됨) => dns-name은 소문자로 고유하게 작성해야 함	az network public-ip create --resource-group KBCN_dheo --location koreacentral --name peer4-pip4 --dns-name publicdnspeer4 --allocation-method static --idle-timeout 4 
8. 서브넷과 연결된 NIC 만들기(건너뛰어도 됨)	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
9. VM 생성해 복사된 Disk와 연결	az vm create --resource-group KBCN_dheo --name peer4 --nics peer4-nic4 --size Standard_D2s_v3 --os-type Linux --attach-os-disk peer4_OsDisk 

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

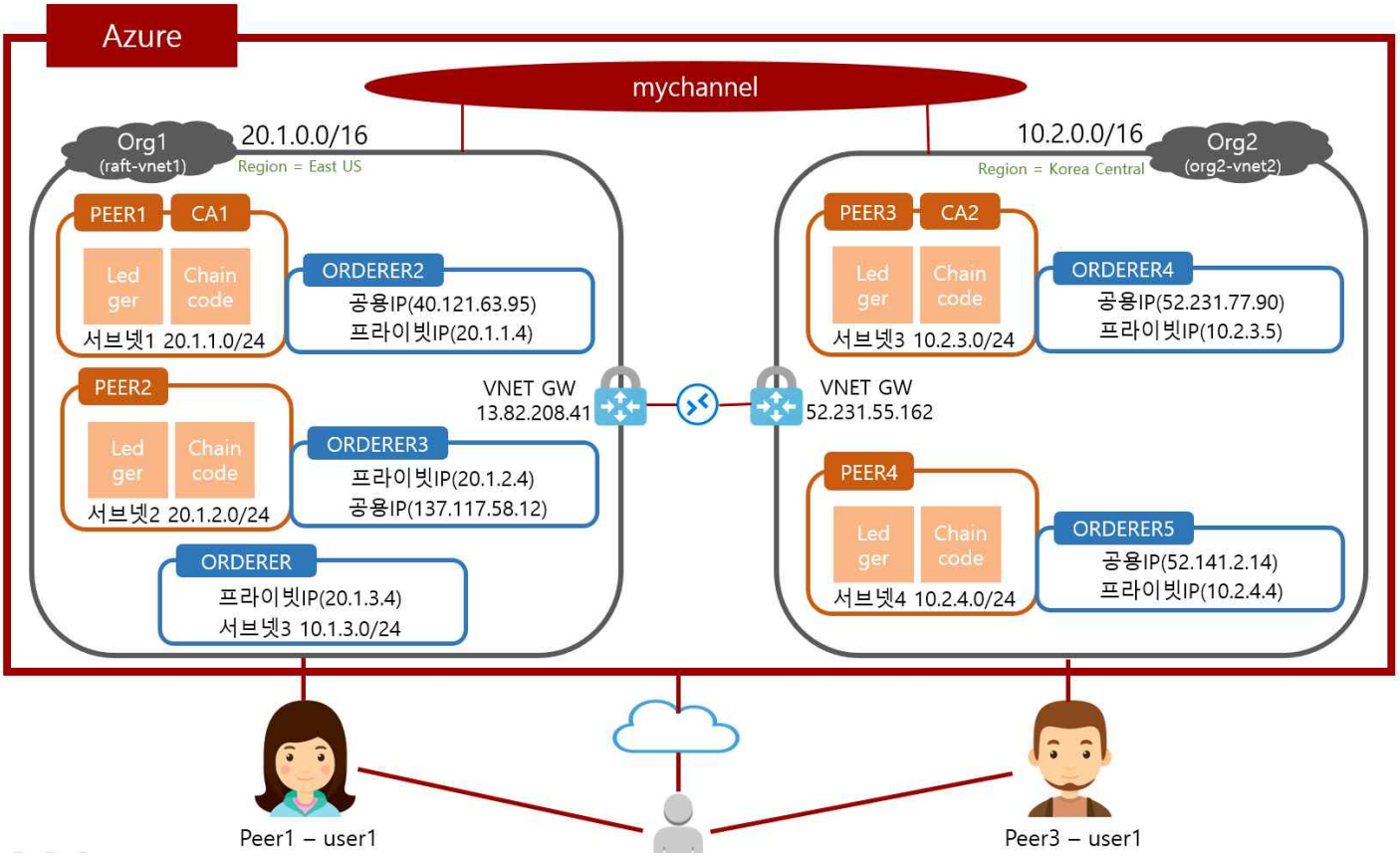
  위 명령어를 입력해주면 0(no)로 바뀌어 ping에 대해 응답하게 됨.

  윈도우 powshell에서는 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-howto-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)

2. docker swarm 만들기

* 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 192.168.56.10
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-1-3bhqk7in4empnjsdho9l5odpc0x24hzswmc5jh13gfmf94u03k-0y9l8f24z8s8w8vfd0hsip9n9 192.168.56.10:2377

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-1-1zqjcs1f63m3um8q9xrk53qbbczz9b6byjov4l9qrykconvp9z-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-1-1zqjcs1f63m3um8q9xrk53qbbczz9b6byjo
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 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

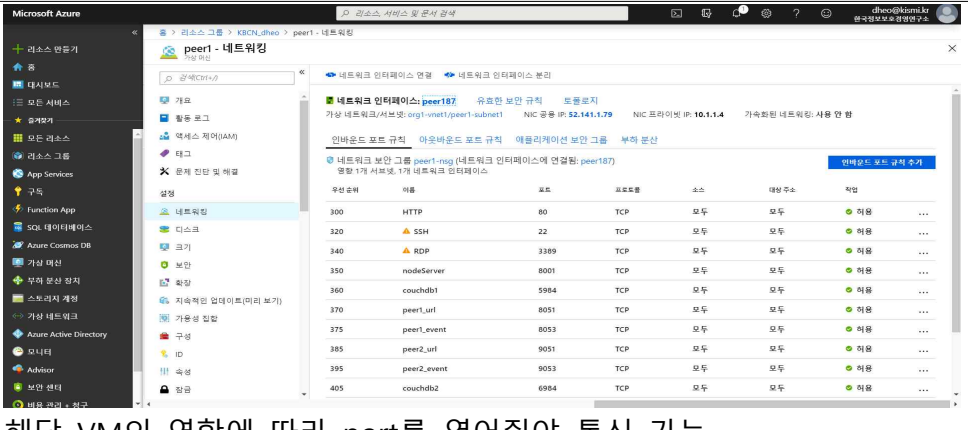
$ docker node ls

```
root@peer1EastUS:~/kismi_blockchain/sdk# docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
7ekunth6t5zj5cim85hj6hgnc     ordererEastUS     Ready     Active         Reachable        19.03.1
l87h0qvczz0ff6lwolq0p9jhc *   peer1EastUS       Ready     Active         Leader           19.03.1
x38713nfi0gb03w9jv7r38b88     peer2EastUS       Ready     Active         Reachable        19.03.1
xor0yk1kuc934ufu5ftdmzn2p     peer3             Ready     Active         Reachable        18.09.7
ku2kwaso3lu7ui909tbeaqayp     peer4             Ready     Active         Reachable        18.09.7
```

$ docker network ls

```
root@peer1EastUS:~/kismi_blockchain/sdk# 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
veq85dt9qa42        kismi_bdms          overlay             swarm
e35100d2ed88        none                null                local
```

3. VM 네트워크 보안규칙 설정

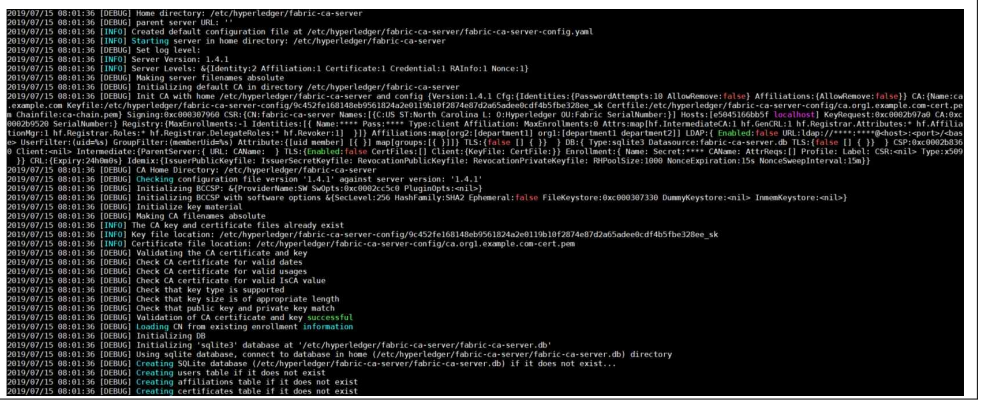


해당 VM의 역할에 따라 port를 열어줘야 통신 가능

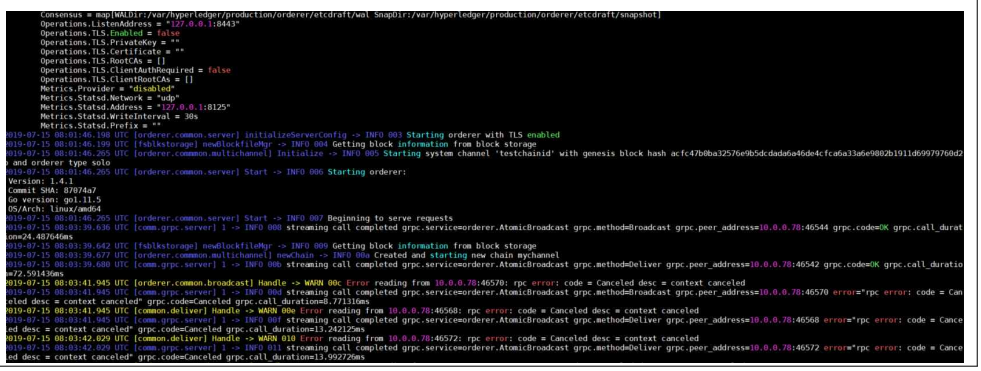
3. docker swarm을 활용해 멀티노드 구성하기(이어서)

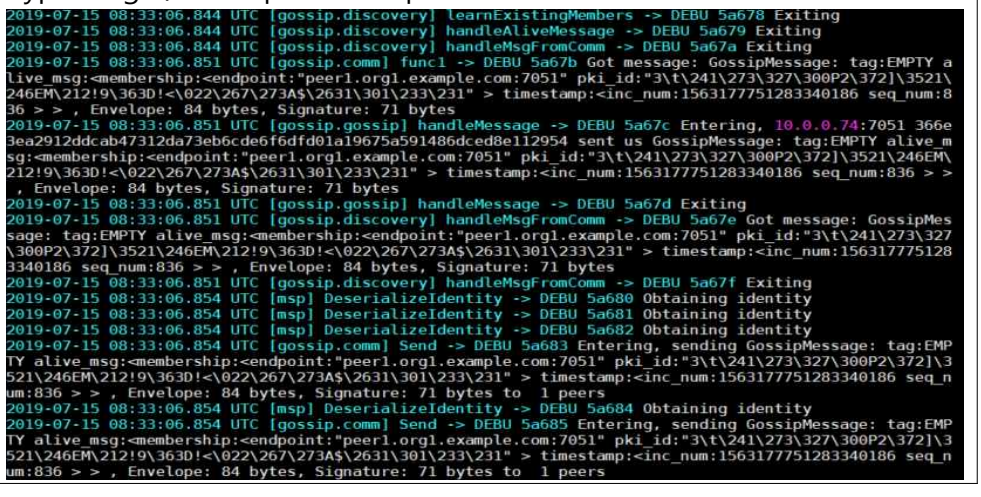
4. MSP 생성 및 배포 (orderer, peer 키 생성)	cryptogen generate --config=./crypto-config.yaml
5. genesisblock 생성 (crypto-config 디렉토리와, configtx.yaml 파일이 있는 곳에서 진행) => -profile 옵션은 변경 가능	configtxgen -profile SampleMultiNodeEtcdRaft -outputBlock genesis.block --channelID mychannel
6. channel 설정 (채널 구축을 위한 트랜잭션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

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

9. CA 서버 구동	docker run -d --rm -it --network="kismi_bdms" --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_TLS_ENABLE=true -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/01d98e1a61f3588c1abd0d35b615394403351b78d8f0601e20586ec81aecec0c_sk -v $(pwd)/crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms hyperledger/fabric-ca:1.4.2 sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/01d98e1a61f3588c1abd0d35b615394403351b78d8f0601e20586ec81aecec0c_sk -b admin:adminpw -d' 

3. docker swarm을 활용해 멀티노드 구성하기(이어서)

10. Orderer 구동	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.genesis.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/server.key -e ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt -e ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt] -e ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE=/var/hyperledger/orderer/tls/server.crt -e ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY=/var/hyperledger/orderer/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/orderer.genesis.block -v $(pwd)/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hyperledger/orderer/msp -v $(pwd)/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/:/var/hyperledger/orderer/tls -w /opt/gopath/src/github.com/hyperledger/fabric hyperledger/fabric-orderer:1.4.2 orderer . 

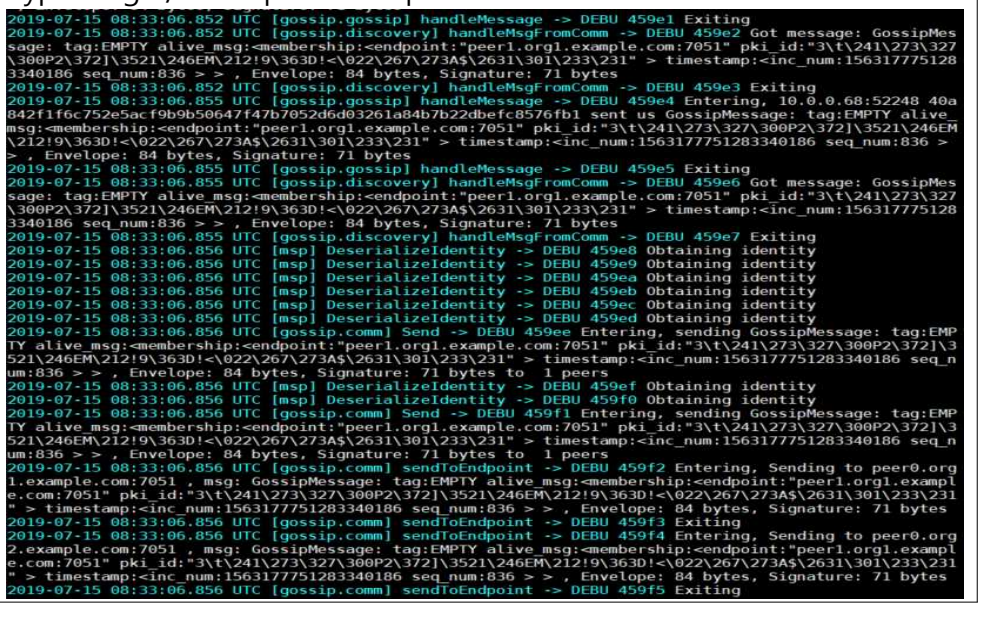
11. CouchDB1 및 Peer1 구동	docker run -d --rm -it --network="kismi_bdms" --name couchdb1 -p 5984:5984 -e COUCHDB_USER= -e COUCHDB_PASSWORD= -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms hyperledger/fabric-couchdb docker run -d --rm -it --link orderer.example.com:orderer.example.com --network="kismi_bdms" --name peer0.org1.example.com -p 8051:7051 -p 8053:7053 -e CORE_LEDGER_STATE_STATEDATABASE=CouchDB -e CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984 -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:7050 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051 -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/peer0.org1.example.com/msp:/etc/hyperledger/fabric/msp -v $(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer0.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 

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=couchdb2:5984 ₩ -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:7050 ₩ -e CORE_PEER_GOSSIP_IGNORESECURITY=true ₩ -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms ₩ -e CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:7051 ₩ -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 "$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp":/etc/hyperledger/fabric/msp ₩ -v "$(pwd)/crypto-config/peerOrganizations/org1.example.com/peers/peer1.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=couchdb3:5984 -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:7050 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org2.example.com:7051 -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/peer0.org2.example.com/msp:/etc/hyperledger/fabric/msp -v $(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer0.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

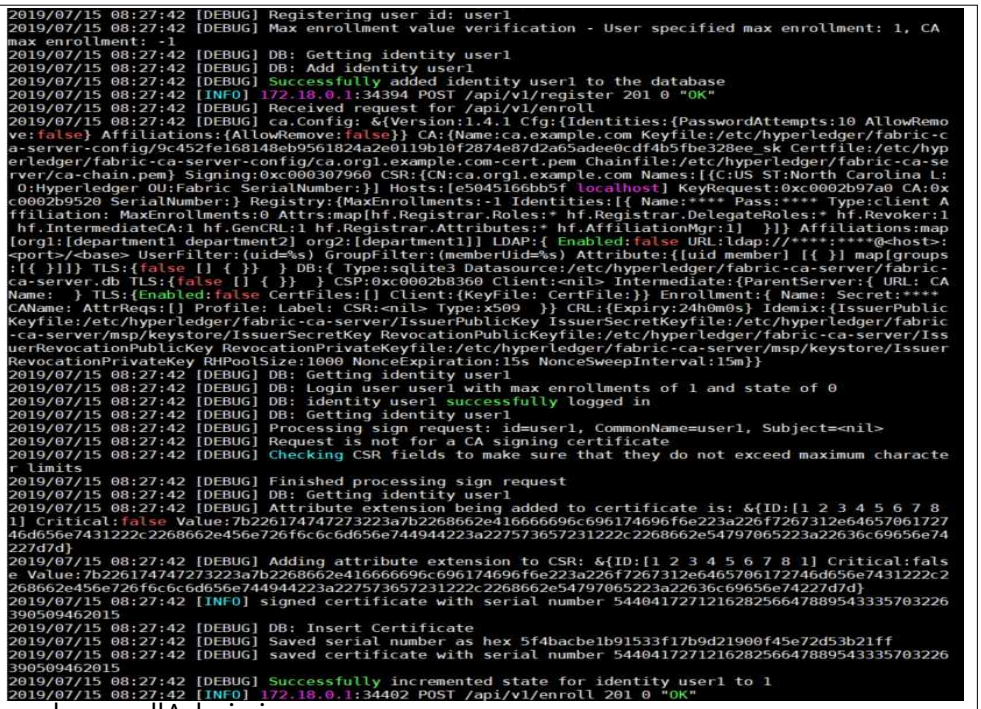
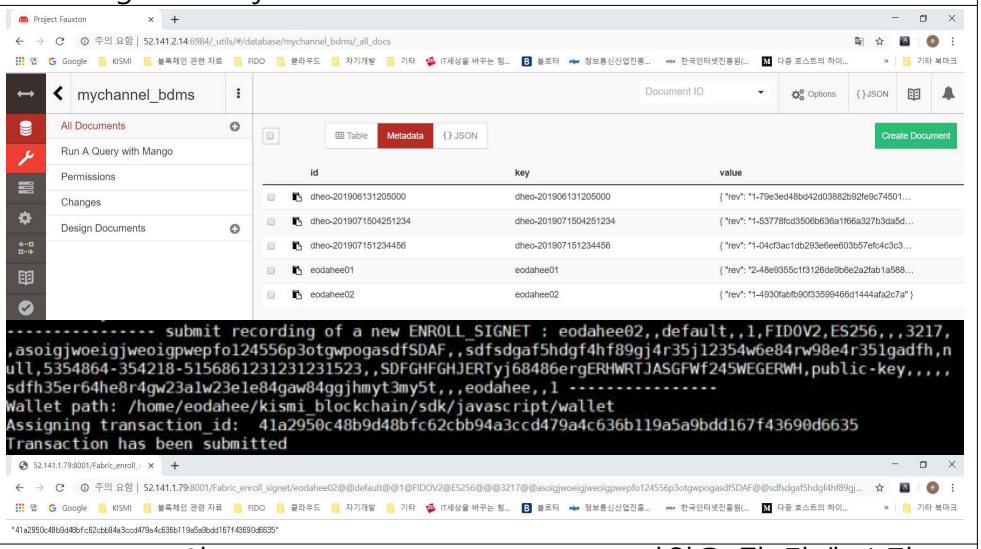
3. docker swarm을 활용해 멀티노드 구성하기(이어서)

14. CouchDB4 및 Peer4 구동	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=couchdb4:5984 -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:7050 -e CORE_PEER_GOSSIP_IGNORESECURITY=true -e CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=kismi_bdms -e CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org2.example.com:7051 -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/peer1.org2.example.com/msp:/etc/hyperledger/fabric/msp -v $(pwd)/crypto-config/peerOrganizations/org2.example.com/peers/peer1.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)

15. CLI 구동 및 channel create, join, chaincode install, instantiated 등 작업 (완료되면 맨 뒤에 -c 이후만 지우고 /bin/bash까지 실행하면 cli 컨테이너로 입성)	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/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/server.crt -e CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/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/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.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/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@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/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:1.4.2 /bin/bash -c './scripts/script.sh' 

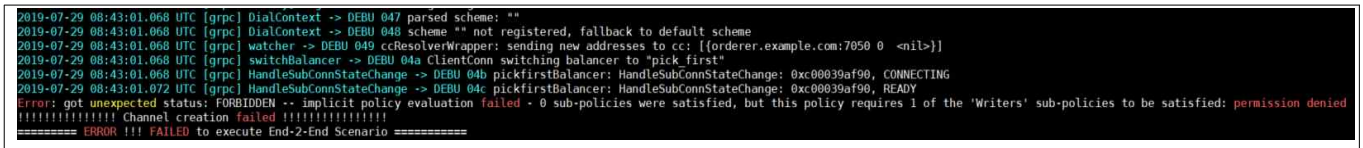
3. docker swarm을 활용해 멀티노드 구성하기(이어서)

16. admin, user 등록을 통해 sdk를 활용한 통신	 node enrollAdmin.js node registerUser.js
17. couchDB 및 web 확인	
18. 주의 및 참고사항	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_peer00_orderer00.sh 파일을 만들어 구동

4. 네트워크 구성 간 발생한 오류들

① 권한 오류 문제



[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
              ClientTLSCert: crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
              ServerTLSCert: crypto-config/ordererOrganizations/example.com/orderers/orderer.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
            - Host: orderer4.example.com
              Port: 10050
              ClientTLSCert: crypto-config/ordererOrganizations/example.com/orderers/orderer4.example.com/tls/server.crt
              ServerTLSCert: crypto-config/ordererOrganizations/example.com/orderers/orderer4.example.com/tls/server.crt
            - Host: orderer5.example.com
              Port: 11050
              ClientTLSCert: crypto-config/ordererOrganizations/example.com/orderers/orderer5.example.com/tls/server.crt
              ServerTLSCert: 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 implict authen 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                                NAMES                COMMAND                  CREATED          STATUS          PORTS
5db50fe15f86   hyperledger/fabric-ca:1.4.1          ca.org2.example.com  "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)
http://blog.naver.com/PostView.nhn?blogId=mk0278&logNo=220253821&categoryNo=0&parentCategoryNo=0&viewDate=&currentPage=1&postListTopCurrentPage=1&from=postView
- 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
```