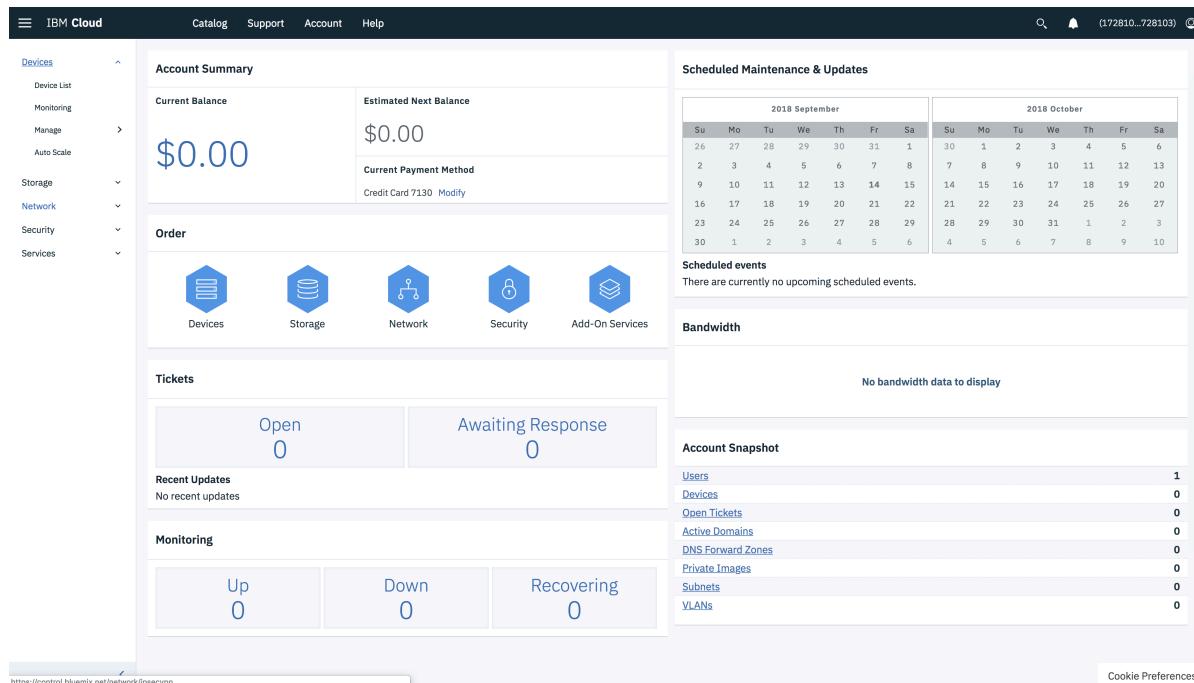


Hyperledger Fabric 쿠버네티스에 올리기

개발 환경 설정

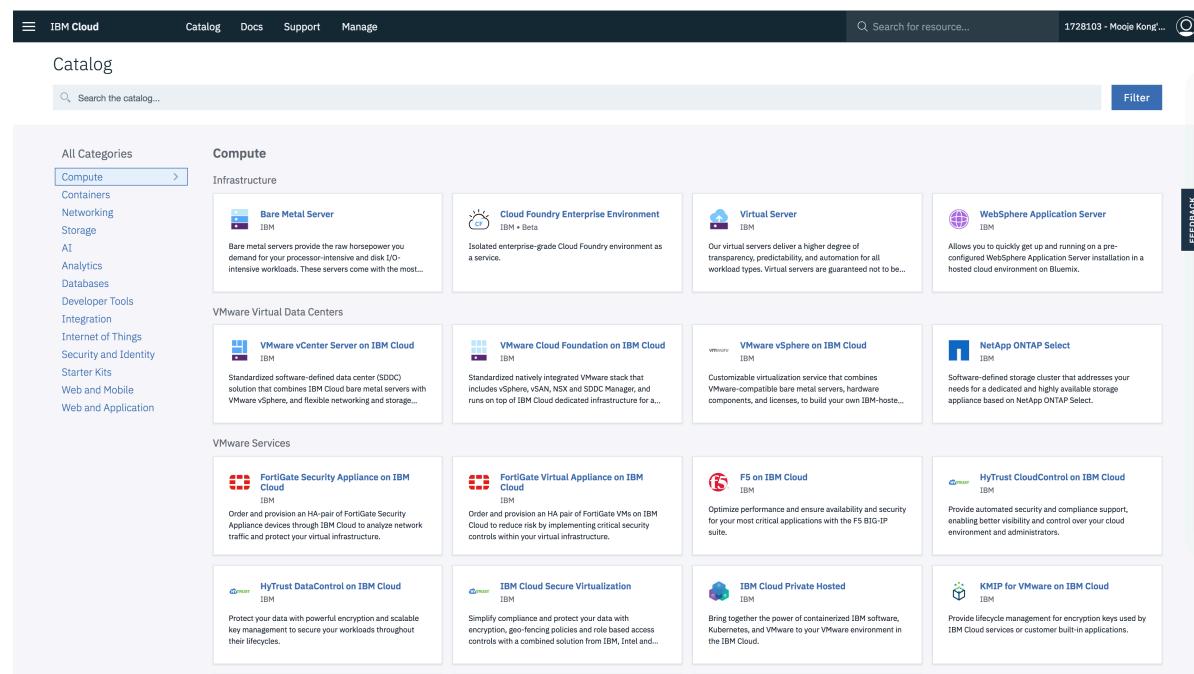
개발용 VM 생성

<https://www.bluemix.net/>에 로그인하여 메뉴에서 “Infrastructure > Devices” 선택



The screenshot shows the IBM Cloud Infrastructure > Devices interface. The left sidebar is collapsed, showing the 'Devices' section. The main area has three main sections: 'Account Summary' (Current Balance: \$0.00, Estimated Next Balance: \$0.00, Current Payment Method: Credit Card 7130 Modify), 'Scheduled Maintenance & Updates' (two calendar grids for September and October 2018), and 'Bandwidth' (No bandwidth data to display). Below these are 'Tickets' (Open: 0, Awaiting Response: 0) and 'Monitoring' (Up: 0, Down: 0, Recovering: 0). The bottom right corner shows 'Cookie Preferences' and the URL 'https://control.bluemix.net/network/pssecvpn'.

Compute 리스트에서 “Virtual Server” 선택



The screenshot shows the IBM Cloud Catalog interface. The left sidebar shows 'Compute' selected under 'All Categories'. The main area is the 'Compute' section, which includes 'Infrastructure' (Bare Metal Server, Cloud Foundry Enterprise Environment, Virtual Server, WebSphere Application Server), 'VMware Virtual Data Centers' (VMware vCenter Server on IBM Cloud, VMware Cloud Foundation on IBM Cloud, VMware vSphere on IBM Cloud, NetApp ONTAP Select), 'VMware Services' (FortiGate Security Appliance on IBM Cloud, FortiGate Virtual Appliance on IBM Cloud, F5 on IBM Cloud, HyTrust CloudControl on IBM Cloud), and 'Cloud Services' (IBM Cloud Secure Virtualization, IBM Cloud Private Hosted, KMIP for VMware on IBM Cloud). Each service card provides a brief description and a 'View Details' button.

Public Virtual Server 선택 후 빌링 타입을 "Hourly"로 변경 후 화면 오른쪽 하단의 "Create" 버튼 클릭

Hostname은 적절히 입력하고 Image는 Ubuntu 16.04를 선택

나머지 항목들 점검(기본값) 후 화면 오른쪽에서 “Order Summary” 확인 후 VM 생성 완료

The screenshot shows the IBM Cloud interface for creating a new VM. The 'Order Summary' section on the right details the configuration:

- VM Type:** 1 Virtual Server (Public)
- Compute:** Compute C1.x4
- Memory:** 1 vCPU, 1 GB RAM
- Storage:** 100 GB (100 GB SSD)
- OS:** Ubuntu 16.04 LTS Xenial Xerus Minimal Install (64-bit)
- Networking:** 100 Mbps Public & Private Network Uplinks (\$0.00)
- Block Storage:** 25 GB (SAN) (\$0)
- Network Interface:** 1 Network Interface (\$0.00)
- Total due per hour***: \$0.04 (estimated)

Below the summary, there are notes about price based on average usage and public bandwidth charged per GB. A checkbox for accepting third-party agreements is present, along with links to the Ubuntu license and IBM Cloud Sales.

The screenshot shows the 'Devices' page in the IBM Cloud interface. The newly created VM is listed:

DEVICE NAME	DEVICE TYPE	LOCATION	PUBLIC IP	PRIVATE IP	START DATE	ACTIONS
hifcli01.Mooje-Kong-s-Account.cloud	Virtual Server	Dallas 13	169.62.213.16	10.73.229.203	2018-09-13	Actions

VM 생성 후 Devices 화면에서 Host 명을 클릭하여 상세 페이지로 이동 후, 접속을 위한 Public IP 및 계정 정보를 확인하여 터미널 접속

메뉴에서 "Containers" 선택 후 "Create a cluster" 버튼 클릭

Cluster type 을 "Free"로 선택하고 "Create Cluster" 버튼 클릭

클러스터의 “Access” 탭에 설명되어 있는 내용을 기준으로 앞서 생성한 VM에 접속하여
ibmcloud CLI 설치

Clusters / mycluster

mycluster Expires 1ヶ月後 Requested

Access Overview Worker Nodes Worker Pools

Gain access to your cluster

Prerequisites

Download and install a few CLI tools and the IBM Kubernetes Service plug-in.

```
curl -sL https://ibm.biz/idt-installer | bash
```

Gain access to your cluster

1. Log in to your IBM Cloud account.

```
ibmcloud login -a https://api.au-syd.bluemix.net
```

If you have a federated ID, use `ibmcloud login --ss0` to log in to the IBM Cloud CLI.

2. Target the IBM Cloud Container Service region in which you want to work.

```
ibmcloud cs region-set ap-south
```

3. Get the command to set the environment variable and download the Kubernetes configuration files.

```
ibmcloud cs cluster-config mycluster
```

4. Set the `KUBECONFIG` environment variable. Copy the output from the previous command and paste it in your terminal. The command output should look similar to the following.

```
export KUBECONFIG=/Users/$USER/.bluemix/plugins/container-service/clusters/mycluster/kube-config-mel01-mycluster.yml
```

아래와 같이 CLI 설치 완료 확인

```
2. root@hlfcli01: ~ (ssh)

root@hlfcli01:~# ibmcloud login -a https://api.au-syd.bluemix.net
API endpoint: https://api.au-syd.bluemix.net

Email>moojae.kong@gmail.com

Password>
Authenticating...
OK

Targeted account Mooje Kong's Account (fceb206a18a44deab234d39a2019db8e) <-> 1728
103

Targeted resource group Default

API endpoint: https://api.au-syd.bluemix.net
Region: au-syd
User: moojae.kong@gmail.com
Account: Mooje Kong's Account (fceb206a18a44deab234d39a2019db8e) <-> 17
28103
Resource group: Default
CF API endpoint:
Org:
Space:

Tip: If you are managing Cloud Foundry applications and services
- Use 'ibmcloud target --cf' to target Cloud Foundry org/space interactively, or
use 'ibmcloud target --cf-api ENDPOINT -o ORG -s SPACE' to target the org/space.
- Use 'ibmcloud cf' if you want to run the Cloud Foundry CLI with current IBM Cloud
CLI context.

root@hlfcli01:~#
```

다음의 명령을 통해 클러스터에 로그인

```
root@hlfcli01:~# ibmcloud cs region-set ap-south
If you have clusters that run Kubernetes versions 1.5, 1.7 or 1.8, update them now to continue receiving important security updates and support. Kubernetes version 1.8 is deprecated and will be unsupported 22 Sept 2018. Versions 1.5 and 1.7 are already unsupported. For more information and update actions, see <https://ibm.biz/iks-versions>
```

OK

```
root@hlfcli01:~# ibmcloud cs cluster-config mycluster
```

OK

The configuration for **mycluster** was downloaded successfully. Export environment variables to start using Kubernetes.

```
export KUBECONFIG=/root/.bluemix/plugins/container-service/clusters/mycluster/kube-config-mel01-mycluster.yml
```

모든 명령 실행 후 KUBECONFIG 환경 변수를 설정 후 "kubectl get nodes" 명령을 통해 클러스터 접속 확인

```
root@hlfcli01:~# export KUBECONFIG=/root/.bluemix/plugins/container-service/clusters/mycluster/kube-config-mel01-mycluster.yml
root@hlfcli01:~# kubectl get nodes
NAME           STATUS    ROLES     AGE      VERSION
10.118.181.136   Ready    <none>    1m       v1.10.7+IKS
```

Hyperledger Fabric 배포

Github에서 소스를 받기 위해 git 패키지를 설치

```
> apt install -y git
```

설치 이후 특정 디렉토리에 소스 다운로드

```
> git clone https://github.com/mjkong/mymarket.git
```

```
2. root@hlfcli01: ~ (ssh)
root@hlfcli01:~# git clone https://github.com/mjkong/mymarket.git
Cloning into 'mymarket'...
remote: Counting objects: 433, done.
remote: Compressing objects: 100% (156/156), done.
remote: Total 433 (delta 24), reused 179 (delta 22), pack-reused 247
Receiving objects: 100% (433/433), 112.25 KiB | 1.84 MiB/s, done.
Resolving deltas: 100% (84/84), done.
root@hlfcli01:~#
```

IBM Kubernetes Service에 배포하기 위해서 다음의 경로로 이동

```
> cd ./mymarket/deploy/k8s/kube-config
```

다음의 순서로 배포

1. Persistent Volume 생성

다음의 명령을 실행

```
> kubectl create -f createVolume.yaml
```

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f createVolume.yaml
persistentvolume/shared-pv created
persistentvolumeclaim/shared-pvc created
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl get pv
NAME      CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM
STORAGECLASS  REASON   AGE
shared-pv   1Gi        RWX          Retain        Bound    default/shared-pvc
            3m
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl get pvc
NAME      STATUS      VOLUME      CAPACITY   ACCESS MODES  STORAGECLASS  AGE
shared-pvc  Bound      shared-pv   1Gi        RWX          -          3m
```

2. 아티팩트 설정파일 복사를 위한 Job

다음의 명령을 실행

```
> kubectl create -f createArtifactsJob.yaml
```

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f createArtifactsJob.yaml
job.batch/copyartifacts created
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# k get pod
NAME        READY   STATUS    RESTARTS   AGE
copyartifacts-mtr9n  1/1     Running   0          8s
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl cp ..../artifacts/ copyartifacts-mtr9n:/shared/
kubtctl: command not found
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl cp ..../artifacts/ copyartifacts-mtr9n:/shared/
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# k get po
NAME        READY   STATUS    RESTARTS   AGE
copyartifacts-mtr9n  0/1     Completed  0          1m
```

3. Genesis block 생성

```
> kubectl create -f generateArtifactsJob.yaml
```

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f generateArtifactsJob.yaml
job.batch/utils created
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl get pod
NAME        READY   STATUS    RESTARTS   AGE
copyartifacts-mtr9n  0/1     Completed  0          7m
utils-9zxc2  0/2     Completed  0          11s
```

4. Service 생성

```
> kubectl create -f blockchain-service.yaml
```

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f blockchain-services.yaml
service/mymarket-ca created
service/store1-ca created
service/store2-ca created
service/mymarket-orderer created
service/store1peer0 created
service/store1peer1 created
service/store2peer0 created
service/store2peer1 created
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl get svc
NAME            TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes      ClusterIP   172.21.0.1      <none>          443/TCP          2h
mymarket-ca     NodePort    172.21.74.36    <none>          30054:30054/TCP  27s
mymarket-orderer NodePort    172.21.175.135   <none>          31010:31010/TCP  27s
store1-ca       NodePort    172.21.83.181    <none>          30154:30154/TCP  27s
store1peer0     NodePort    172.21.188.140   <none>          30110:30110/TCP,30111:30111/TCP  27s
store1peer1     NodePort    172.21.57.3       <none>          30120:30120/TCP,30121:30121/TCP  27s
store2-ca       NodePort    172.21.155.176   <none>          30254:30254/TCP  27s
store2peer0     NodePort    172.21.158.90    <none>          30210:30210/TCP,30211:30211/TCP  26s
store2peer1     NodePort    172.21.167.0       <none>          30220:30220/TCP,30221:30221/TCP  26s
```

5. 컨테이너(POD) 생성

> kubectl create -f peerDeployment.yaml

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f peersDeployment.yaml
deployment.extensions/mymarket-orderer created
deployment.extensions/store1-ca created
deployment.extensions/store2-ca created
deployment.extensions/mymarket-ca created
deployment.extensions/store1peer0 created
deployment.extensions/store1peer1 created
deployment.extensions/store2peer0 created
deployment.extensions/store2peer1 created
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# k get po
NAME            READY   STATUS    RESTARTS   AGE
copyartifacts-mtr9n   0/1    Completed  0          18m
mymarket-ca-66d7fd8ffc-szgl5  1/1    Running   0          7m
mymarket-orderer-5d4db59bc5-rtptq  1/1    Running   0          7m
store1-ca-7cdf6c579b-85479   1/1    Running   0          7m
store1peer0-8464479b6f-t7svv  1/1    Running   0          7m
store1peer1-59c878d57c-nkp7v  1/1    Running   0          7m
store2-ca-57c5d44595-pk92j   1/1    Running   0          7m
store2peer0-6f95c88f55-shvc8  1/1    Running   0          7m
store2peer1-7df655bcc9-2qlvn  1/1    Running   0          7m
utils-9zxc2                 0/2    Completed  0          11m
```

6. 채널 아티팩트 및 채널 생성

```
> kubectl create -f create_channel.yaml
```

NAME	READY	STATUS	RESTARTS	AGE
copyartifacts-mtr9n	0/1	Completed	0	23m
createchanneltx-k1xmw	0/2	Completed	0	1m
mymarket-ca-66d7fd8ffc-szgl5	1/1	Running	0	11m
mymarket-orderer-5d4db59bc5-rtptq	1/1	Running	0	11m
store1-ca-7cdf6c579b-85479	1/1	Running	0	11m
store1peer0-8464479b6f-t7svv	1/1	Running	0	11m
store1peer1-59c878d57c-nkp7v	1/1	Running	0	11m
store2-ca-57c5d44595-pk92j	1/1	Running	0	11m
store2peer0-6f95c88f55-shvc8	1/1	Running	0	11m
store2peer1-7df655bcc9-2qlvn	1/1	Running	0	11m
utils-9zxc2	0/2	Completed	0	16m

7. 피어 채널 조인

```
> kubectl create -f join_channel.yaml
```

root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl create -f join_channel.yaml				
job.batch/joinchannel created				
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl get pod				
NAME	READY	STATUS	RESTARTS	AGE
copyartifacts-mtr9n	0/1	Completed	0	28m
createchanneltx-k1xmw	0/2	Completed	0	6m
joinchannel-ps9sb	0/4	Completed	0	2m
mymarket-ca-66d7fd8ffc-szgl5	1/1	Running	0	16m
mymarket-orderer-5d4db59bc5-rtptq	1/1	Running	0	16m
store1-ca-7cdf6c579b-85479	1/1	Running	0	16m
store1peer0-8464479b6f-t7svv	1/1	Running	0	16m
store1peer1-59c878d57c-nkp7v	1/1	Running	0	16m
store2-ca-57c5d44595-pk92j	1/1	Running	0	16m
store2peer0-6f95c88f55-shvc8	1/1	Running	0	16m
store2peer1-7df655bcc9-2qlvn	1/1	Running	0	16m
utils-9zxc2	0/2	Completed	0	21m

8. 체인코드 인스톨

```
> kubectl create -f chaincode_install.yaml
```

NAME	READY	STATUS	RESTARTS	AGE
chaincodeinstall-67988	0/4	Completed	0	33s
copyartifacts-mtr9n	0/1	Completed	0	30m
createchanneltx-klxmw	0/2	Completed	0	8m
joinchannel-ps9sb	0/4	Completed	0	4m
mymarket-ca-66d7fd8ffc-szgl5	1/1	Running	0	18m
mymarket-orderer-5d4db59bc5-rtptq	1/1	Running	0	18m
store1-ca-7cdf6c579b-85479	1/1	Running	0	18m
store1peer0-8464479b6f-t7svv	1/1	Running	0	18m
store1peer1-59c878d57c-nkp7v	1/1	Running	0	18m
store2-ca-57c5d44595-pk92j	1/1	Running	0	18m
store2peer0-6f95c88f55-shvc8	1/1	Running	0	18m
store2peer1-7df655bcc9-2qlvn	1/1	Running	0	18m
utils-9zxc2	0/2	Completed	0	23m

9. 체인코드 초기화

```
> kubectl create -f chaincode_instantiate.yaml
```

NAME	READY	STATUS	RESTARTS	AGE
chaincodeinstall-67988	0/4	Completed	0	1m
chaincodeinstantiate-jlbbt	0/1	Completed	0	19s
copyartifacts-mtr9n	0/1	Completed	0	31m
createchanneltx-klxmw	0/2	Completed	0	9m
joinchannel-ps9sb	0/4	Completed	0	5m
mymarket-ca-66d7fd8ffc-szgl5	1/1	Running	0	19m
mymarket-orderer-5d4db59bc5-rtptq	1/1	Running	0	19m
store1-ca-7cdf6c579b-85479	1/1	Running	0	19m
store1peer0-8464479b6f-t7svv	1/1	Running	0	19m
store1peer1-59c878d57c-nkp7v	1/1	Running	0	19m
store2-ca-57c5d44595-pk92j	1/1	Running	0	19m
store2peer0-6f95c88f55-shvc8	1/1	Running	0	19m
store2peer1-7df655bcc9-2qlvn	1/1	Running	0	19m
utils-9zxc2	0/2	Completed	0	24m

10. 테스트

Hyperledger Fabric 모든 환경이 배포되었으면 각 피어에 접속하여 체인코드 호출을 통해서 정상 동작을 확인

- store1peer0

```
> kubectl exec -it $(kubectl get pod -l name=store1peer0 --output=jsonpath='{.items..metadata.name}') bash
```

```
root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl exec -it $(kubectl get pod -l name=store1peer0 --output=jsonpath='{.items..metadata.name}') bash
root@store1peer0-8464479b6f-t7svv:/#
```

```
> peer chaincode invoke -o mymarket-orderer:31010 -C mymarketchannel -n mycc -c '{"Args":["registProducts","mjcar","1","mj"]}'
```

```

2018-09-14 08:02:26.076 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> DEBU 04e ESCC invoke result: version:1 response:<status:200 > payload:"\n \242\211\231\263\025\274`\\344\300\$M\240\206r\351\200\264\310\032~L\360E\213\355\63\273\232\022\225\001\n\200\001\022\024\004lsc\022\014\n\n\n\004mycc\022\002\010\001\022h\n\004mycc\022`\n\013\n\tlatestKey\032.\n\003PD0\032`{\"name\":\"mjcar\",\"qty\":\"1\"},\"owner\":\"mj\"\}`\032!\n\tlatestKey\032\024`{\"Key\":\"PD\",\"Idx\":0}\032\003\010\310\001`\"\\013\\022\\004mycc\\032\\0031.0\" endorsement:<endorser:>\n\tStore1MSP\022\266\006----BEGIN CERTIFICATE----\nMIICMjCCAdigAwIBAgIRAKxeHWeT5qnjejiAiyic7L0wCgYIKoZIzj0EAwIweTEL\nnMAkGA1UEBhMCVVMxExARBgNVBAgTCKNhbg1mb3JuaWExFjAUBgNVBAcTDVNhbiBG\ncmFuY2lzY28xHDAaBgNVBAoTE3N0b3J1MS5teW1hcmtldC5jb20xHzAdBgNVBAMT\nnFmNhLnN0b3J1MS5teW1hcmtldC5jb20wHhcNMTgwOTE0MDcyNzA0WhcNMjgw0TE\nx\nnMDcyNzA0WjBtMQswCQYDVQGEwJVUzETMBEGA1UECBM\nKQ2FsaWZvcm5pYTEWMBQG\nnA1UEBxMNU2FuIEZyYW5jaXNjbzENMAsGA1UECxMEcGV1cjeiMCAGA1UEAxMzGV1\\ncjAuc3RvcmlUx\nLm15bwFya2V0LmNvbTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IA\nnBMC56D7vuktLt9F+1j5UAvri4Km01Rug+hAkdgFV0FLGdmA\n7JxZrjyXZ9LwjTnc\\nqqGy//yyEfFqmh2o\\4TUo0jTTBLMA4GA1UdDwEB\\wQEawIHgDAMBgNVHRMBAf8E\\nAjAMCsGA1UdIwQk\nMCKAIPHVq31mfxtIBAgrJj8M3bQpfB17eIdI4LI9dfzADn\\nMAoGCCqGSM49BAMCA0gA\nMEUCIQC+nAXqkP7dk+rZT1X5GdWkyz+\n\nnBMJ1BWWU0Uqk\\n6yb\\wQIgPpXtHnVGMP451JG1WEbmkCeXgEWl1xGbpvGSuBiwlX4=\n-----END CERTIFICATE-----\n" sig\nature: "0D\\002 .\\257\\2067\\034\\3478\\237k\\305\\300\\302\\312\\313\\377\\216v\\305TX\\357\\r)e:\\331\\227\\226\\205\\232\\372\\002 g^\\272\\325u\\205]8\\263\\2671\\311,\\036\\014\\256K\\347\\260\\032\\3746@\\340\\312\\217\\3163\\272\\301\\354\\313" >
2018-09-14 08:02:26.077 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 04f Chaincode invoke successful. result: status:200

```

> peer chaincode query -C mymarketchannel -n mycc -c

```
'{"Args":["getProductList",""]}'
```

```

2018-09-14 08:04:43.160 UTC [msp/identity] Sign -> DEBU 045 Sign: digest: C5D54FAE5566B66C67CEF08C065
821A0DCE823489F11A6AE0FD7FE4253BE42A1
[{"Key": "PD0", "Record": {"name": "mjcar", "qty": "1", "owner": "mj"}}
root@store1peer0-8464479b6f-t7svv:/# █

```

- store1peer1

> kubectl exec -it \$(kubectl get pod -l name=store1peer1 --output=jsonpath={.items..metadata.name}) bash

```

root@hlfcli01:~/mymarket/deploy/k8s/kube-config# kubectl exec -it $(kubectl get pod -l name=store1peer1 --output=jsonpath={.items..metadata.name}) bash
root@store1peer1-59c878d57c-nkp7v:/# █

```

> peer chaincode query -C mymarketchannel -n mycc -c

```
'{"Args":["getProductList",""]}'
```

```

2018-09-14 08:11:05.100 UTC [msp/identity] Sign -> DEBU 045 Sign: digest: 0E2F6D9B6BB7B24B4DC0A34D0D6
7A7797AA4C4F42CBE459AE74C61BE6F81C565
[{"Key": "PD0", "Record": {"name": "mjcar", "qty": "1", "owner": "mj"}}
root@store1peer1-59c878d57c-nkp7v:/# █

```

store2peer0, store2peer1에 대해서도 반복확인하여 최초 입력한 product 값이 조회 되는지 확인