**Hyperledger Fabric Private data collection**

이 문서는 <https://hyperledger-fabric.readthedocs.io/en/latest/private_data_tutorial.html> 의 내용을 한글화하였습니다.

이 문서는 참여 기관 내의 인증된 피어에 의해 Private Data 를 입력하거나 조회하기위한 기능을 실습하기 위한 자료입니다.

# 컬렉션 정의 JSON 파일 생성

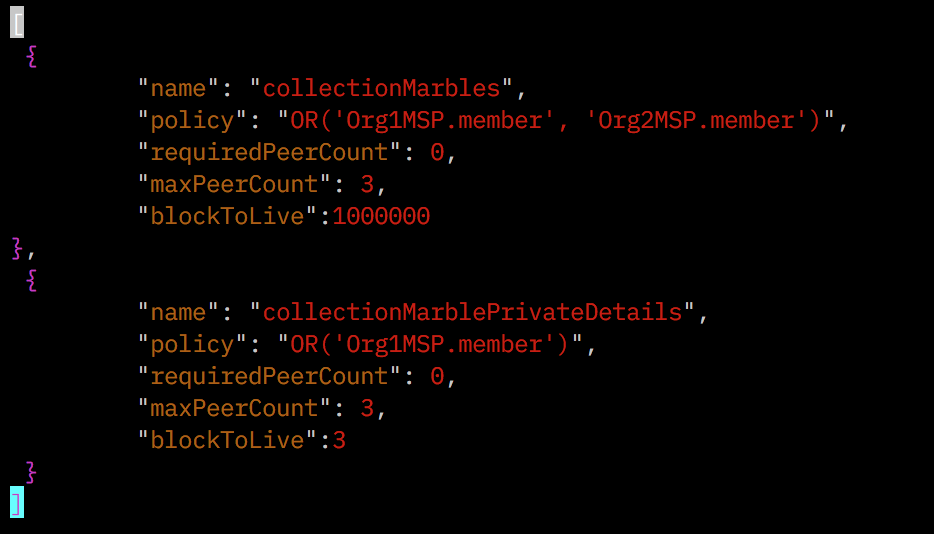
Private Data 사용을 위한 첫 번째 단계는 Private Data 엑세스를 위한 정의 파일을 만드는 것 입니다.

정의 파일은 누가 데이터를 저장할 수 있는지, 얼마나 많은 피어가 데이터를 배포할 수 있는지, 얼마나 많은 피어가 데이터를 공유받아야 하는지, 얼마나 오래 데이터를 보관 할 수 있는지 등을 정의합니다.

이 예제에서는 체인코에서 “PutPrivateData", "GetPrivateData" 등을 어떻게 사용하는지 확인해 보겠습니다.

컬렉션 정의 파일은 다음의 항목으로 구성되어 있습니다.

* name: 컬렉션의 이름
* policy: 컬렉션 데이터를 저장할 수 있는 피어를 정의
* requiredPeerCount: 체인코드에서 endorsement를 유지하기 위해 private data를 저장하는 피어의 수
* maxPeerCount: 데이터 중복성을 위해 현재 endorsing 피어가 데이터를 배포해야하는 다른 피어의 개수. 만약 endorsing 피어가 장애 등 다운되어 있을 때 데이터 요청이 있으면 복제된 데이터를 가지고 있는 피어가 데이터 전달이 가능 함.
* blockToLive: 가격이나 개인 정보와 같은 매우 민감한 정보의 경우 블록 단위로 private 데이터베이스에 데이터를 저장하는 기간을 나타냄. Private 데이터베이스에서 지정된 수의 블록에 대해서 데이터가 존재하고 이후에는 제거되어 네트워크에서 이 데이터를 더 이상 사용할 수 없게 됨. 개인 데이터를 무기한 유지하려면 즉, 개인 데이터를 절대 지우지 않으려면 blockToLive 속성을 0으로 설정



# Private Data 실습

## Fabric Network 시작

기존에 실행중인 도커 컨테이너가 있을 경우 모두 중지 및 삭제합니다.

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

다음의 명령을 통해서 Fabric 네트워크를 실행합니다.

cd /fabric-samples/first-network

./byfn.sh up -c mychannel -s couchdb

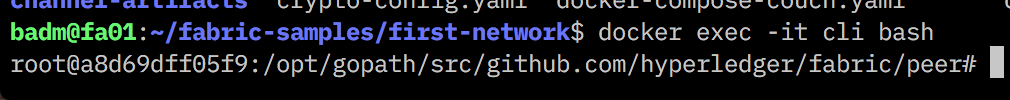
## 체인코드 설치 및 초기화

앞서 실행한 Fabric Network는 네 개의 peer를 가지고 있습니다. 체인코드는 각 피어 별로 모두 설치합니다.

* peer0.org1.example.com
* peer1.org1.example.com
* peer0.org2.example.com
* peer1.org2.example.com

설치에 앞서 cli 컨테이너에 접속합니다. 정상적으로 접속이 되었으면 다음 그림과 같이 프롬프트로 넘어갑니다.

docker exec -it cli bash



각 피어별로 체인코드를 설치합니다.

1. peer0.org1.example.com

peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02\_private/go

1. peer1.org1.example.com

export CORE\_PEER\_ADDRESS=peer1.org1.example.com:7051

peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02\_private/go

1. peer0.org2.example.com

export CORE\_PEER\_LOCALMSPID=Org2MSP

export PEER0\_ORG2\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

export CORE\_PEER\_TLS\_ROOTCERT\_FILE=$PEER0\_ORG2\_CA

export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp

export CORE\_PEER\_ADDRESS=peer1.org1.example.com:7051

peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02\_private/go

1. peer1.org2.example.com

export CORE\_PEER\_ADDRESS=peer1.org2.example.com:7051

peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02\_private/go/

다음으로 체인코드를 인스턴스화 합니다.

export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile $ORDERER\_CA -C mychannel -n marblesp -v 1.0 -c '{"Args":["init"]}' -P "OR('Org1MSP.member','Org2MSP.member')" --collections-config $GOPATH/src/github.com/chaincode/marbles02\_private/collections\_config.json

## Private Data 저장

Org1 조직이 private data detail 을 저장할 권한이 있으므로 CLI에서 Org1 에서 데이터 저장을 위한 체인코드를 호출하겠습니다.

먼저 다음의 명령을 통해 CLI에 Org1 환경변수를 설정합니다.

export CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051

export CORE\_PEER\_LOCALMSPID=Org1MSP

export 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

export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

export PEER0\_ORG1\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org1.example.com/tls/ca.crt

체인코드를 호출합니다.

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marblesp -c '{"Args":["initMarble","marble1","blue","35","tom","99"]}'

## Private Data 조회

Org1에서 marble1에 대한 데이터를 조회해 봅니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarble","marble1"]}'

다시 price 데이터를 조회합니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

Price 정보를 저장할 권한이 없는 Org2에서 다시 조회해 봅니다. 먼저 CLI 환경변수를 Org2 로 전환합니다.

export CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051

export CORE\_PEER\_LOCALMSPID=Org2MSP

export PEER0\_ORG2\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt

export CORE\_PEER\_TLS\_ROOTCERT\_FILE=$PEER0\_ORG2\_CA

export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp

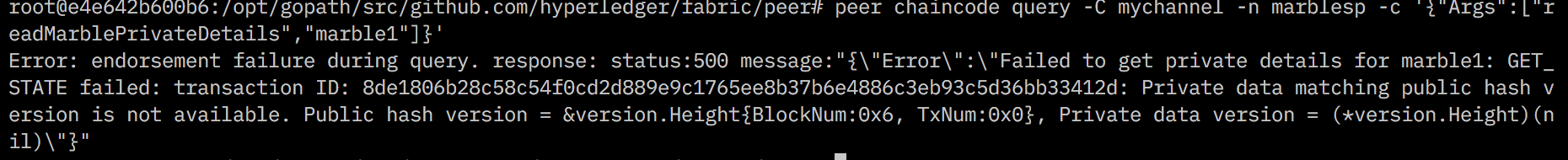
그리고는 marble1의 데이터를 조회합니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarble","marble1"]}'

정상적으로 조회가 되면 price 정보도 조회해 봅니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

다음과 같이 에러 메시지와 함께 권한이 없는 조직에 대한 private data 조회가 되지 않음을 확인 할 수 있습니다.



## Purge private data

Org1의 peer0 의 환경변수를 셋팅합니다.

export CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051

export CORE\_PEER\_LOCALMSPID=Org1MSP

export 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

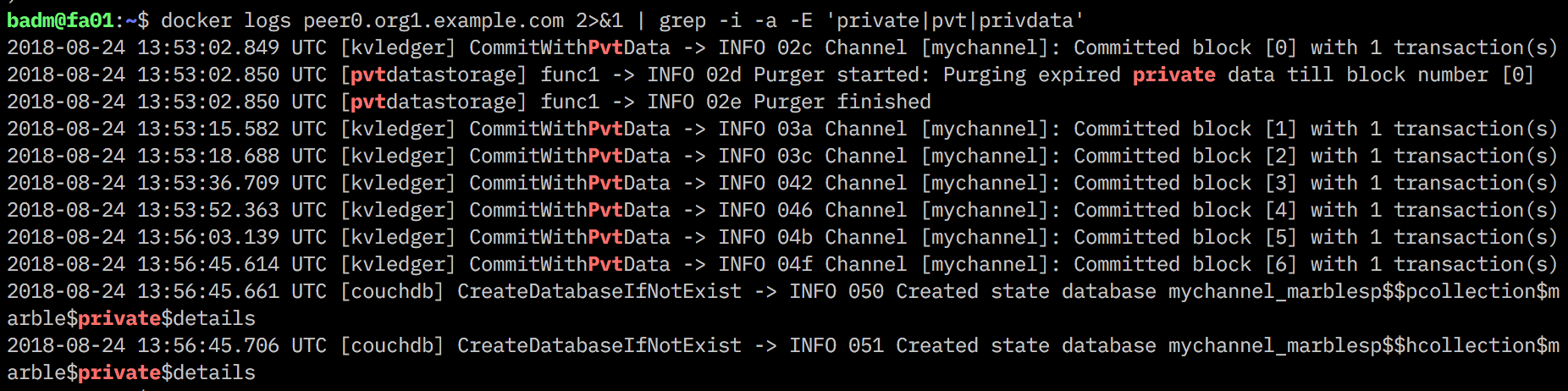
export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

export PEER0\_ORG1\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org1.example.com/tls/ca.crt

다음의 명령을 통해서 블록의 갯수를 모니터링 합니다.

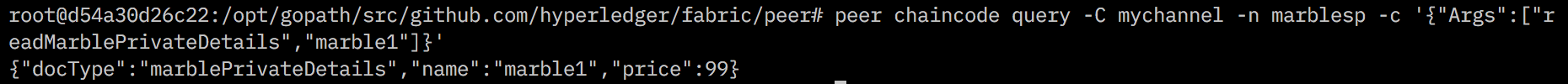
docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'private|pvt|privdata'

현재는 6개의 block height를 가지고 있습니다.



현재 private data 존재 여부를 확인해 봅니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

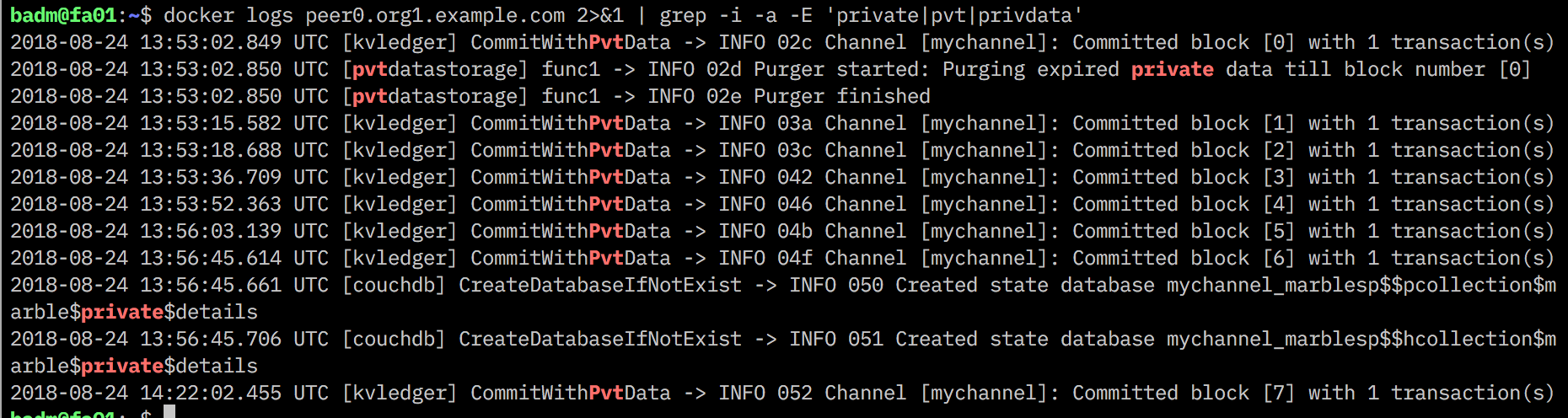


private data 삭제테스트를 위해서 marble2를 입력합니다.

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marblesp -c '{"Args":["initMarble","marble2","blue","35","tom","99"]}'

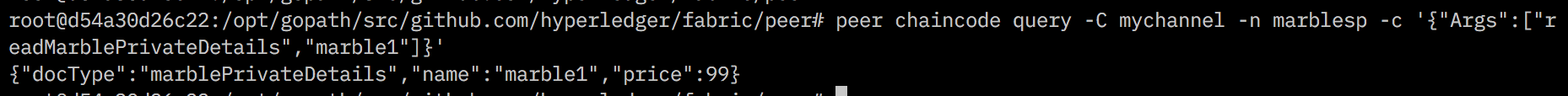
그리고 block 로그를 확인합니다. Blockheight 가 하나 증가됐음을 확인합니다.

docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'private|pvt|privdata'



marble1의 private data가 존재함을 확인합니다.

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

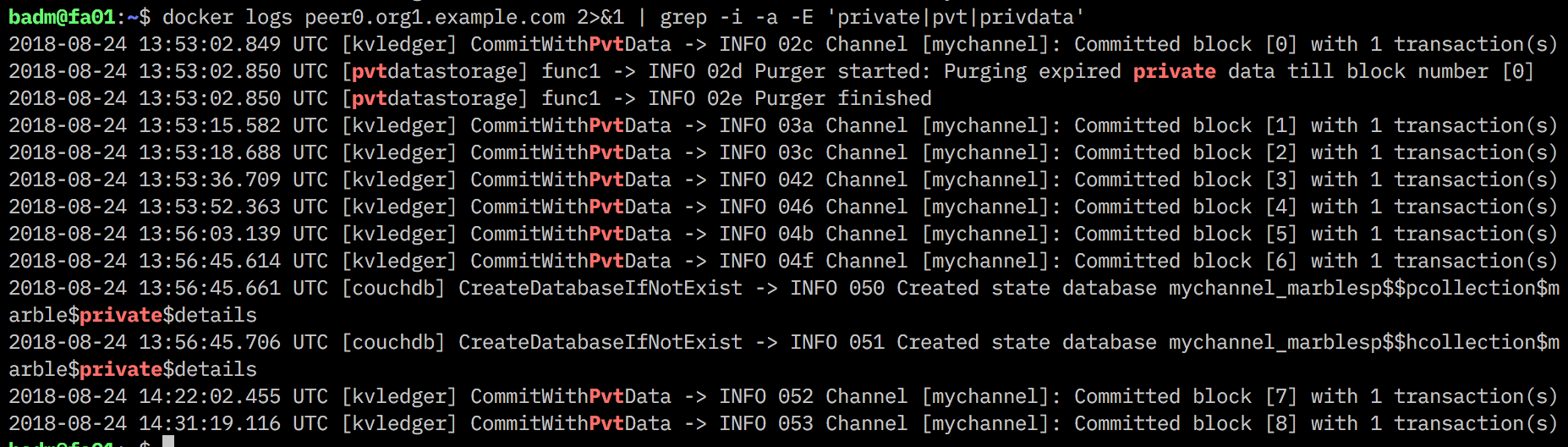


marble2의 소유자를 joe로 변경합니다.

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marblesp -c '{"Args":["transferMarble","marble2","joe"]}'

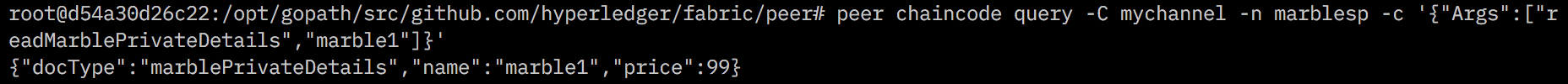
Blockheight 를 확인합니다.

docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'private|pvt|privdata'



다시 marble1의 private data 존재 여부를 확인합니다

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

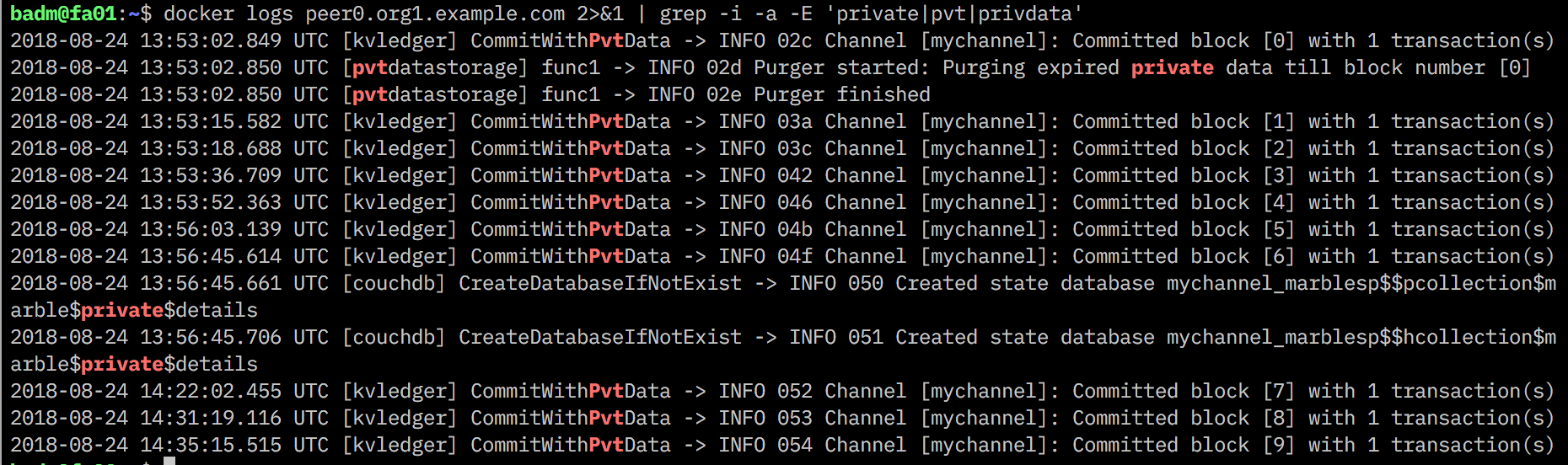


또 새로운 블록 생성을 위해서 marble2의 소유를 tom 으로 변경합니다.

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marblesp -c '{"Args":["transferMarble","marble2","tom"]}'

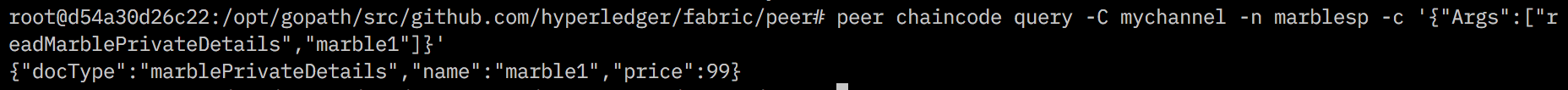
Blockheight 를 확인합니다.

docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'private|pvt|privdata'



다시 marble1의 private data 존재 여부를 확인합니다

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'

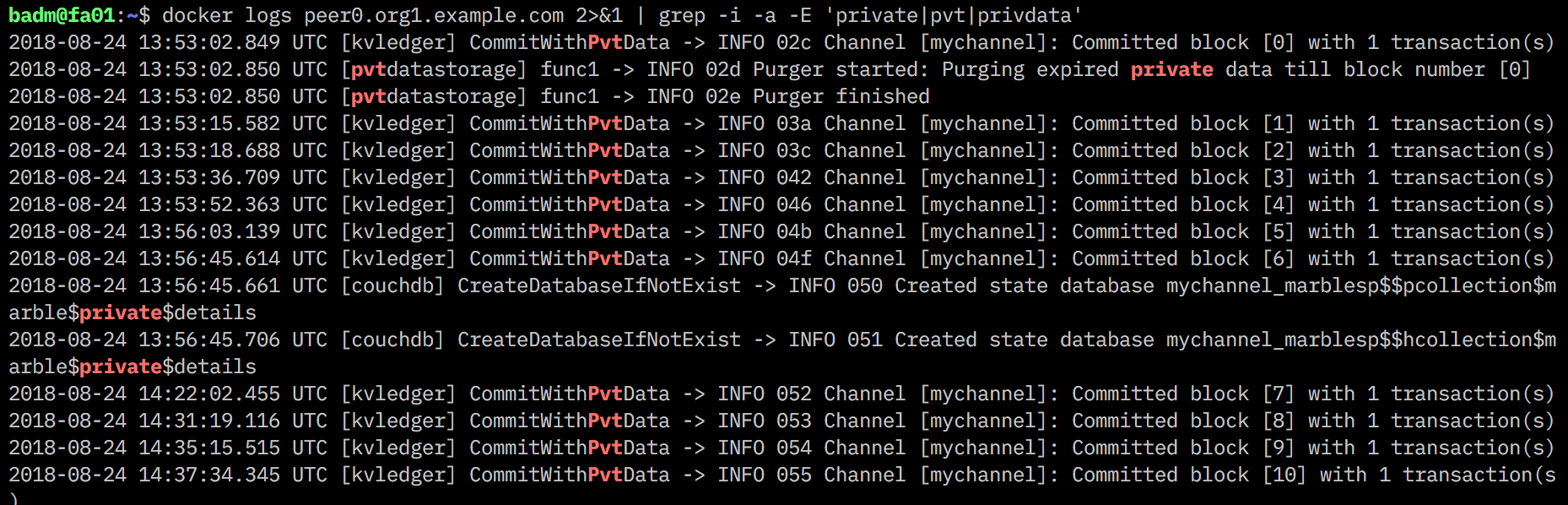


다시 새로운 블록 생성을 위해서 marble2의 소유를 jerry로 변경합니다.

peer chaincode invoke -o orderer.example.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marblesp -c '{"Args":["transferMarble","marble2","jerry"]}'

Blockheight 를 확인합니다.

docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'private|pvt|privdata'



marble1의 private data 존재 여부를 확인합니다

peer chaincode query -C mychannel -n marblesp -c '{"Args":["readMarblePrivateDetails","marble1"]}'



위의 그림에서와 같이 marble1에 대한 데이터가 삭제됨을 확인 할 수 있습니다.