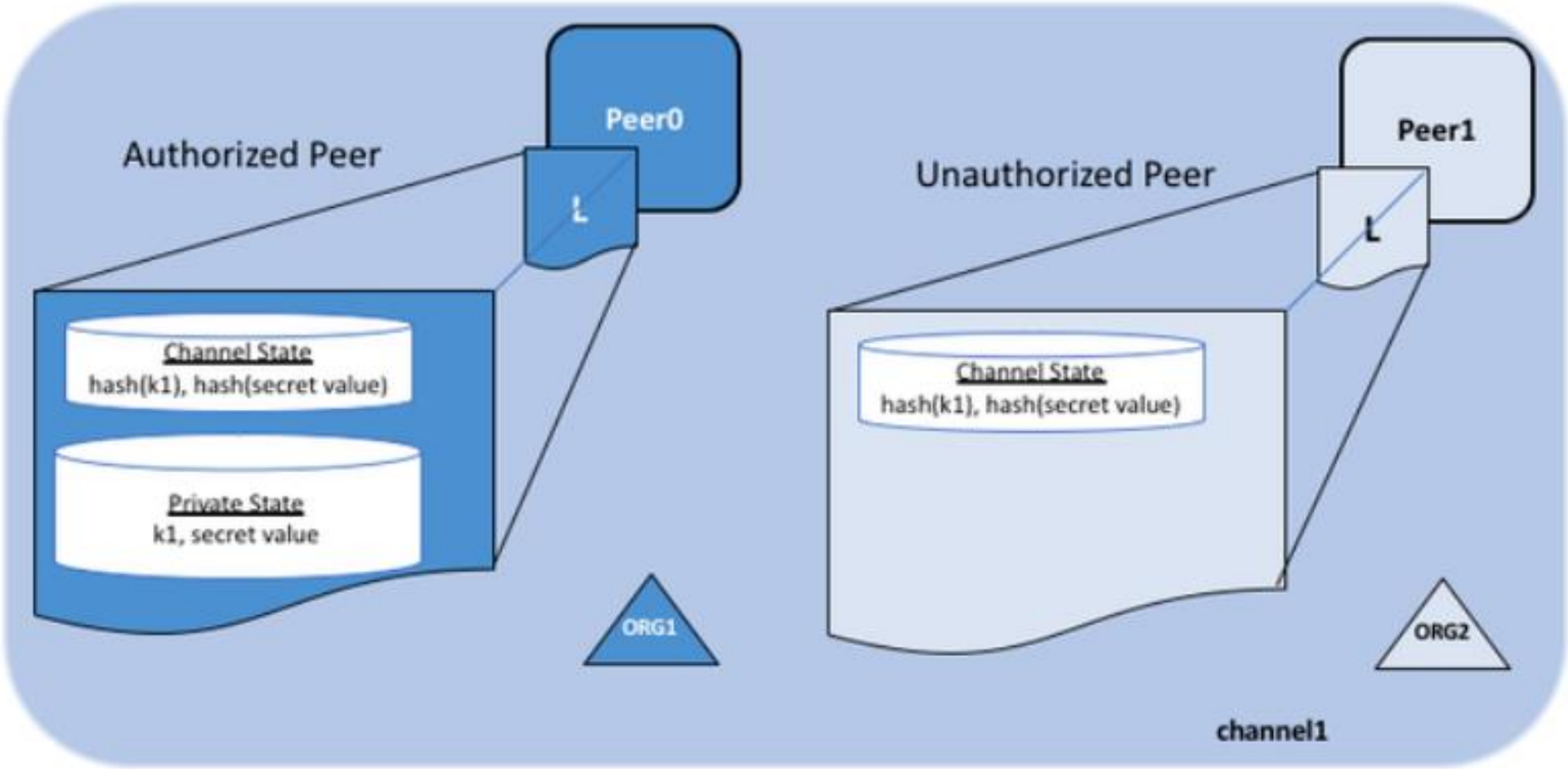
Private Data

- 같은 채널의 다른 조직으로부터 데이터를 보호(비공개)
- 별도의 채널을 생성할 경우 추가적인 관리 비용 발생
  - 체인코드 버전
  - 합의
  - MSP

Private Data Collection

1. The actual private data
2. A hash of that data

Private Data

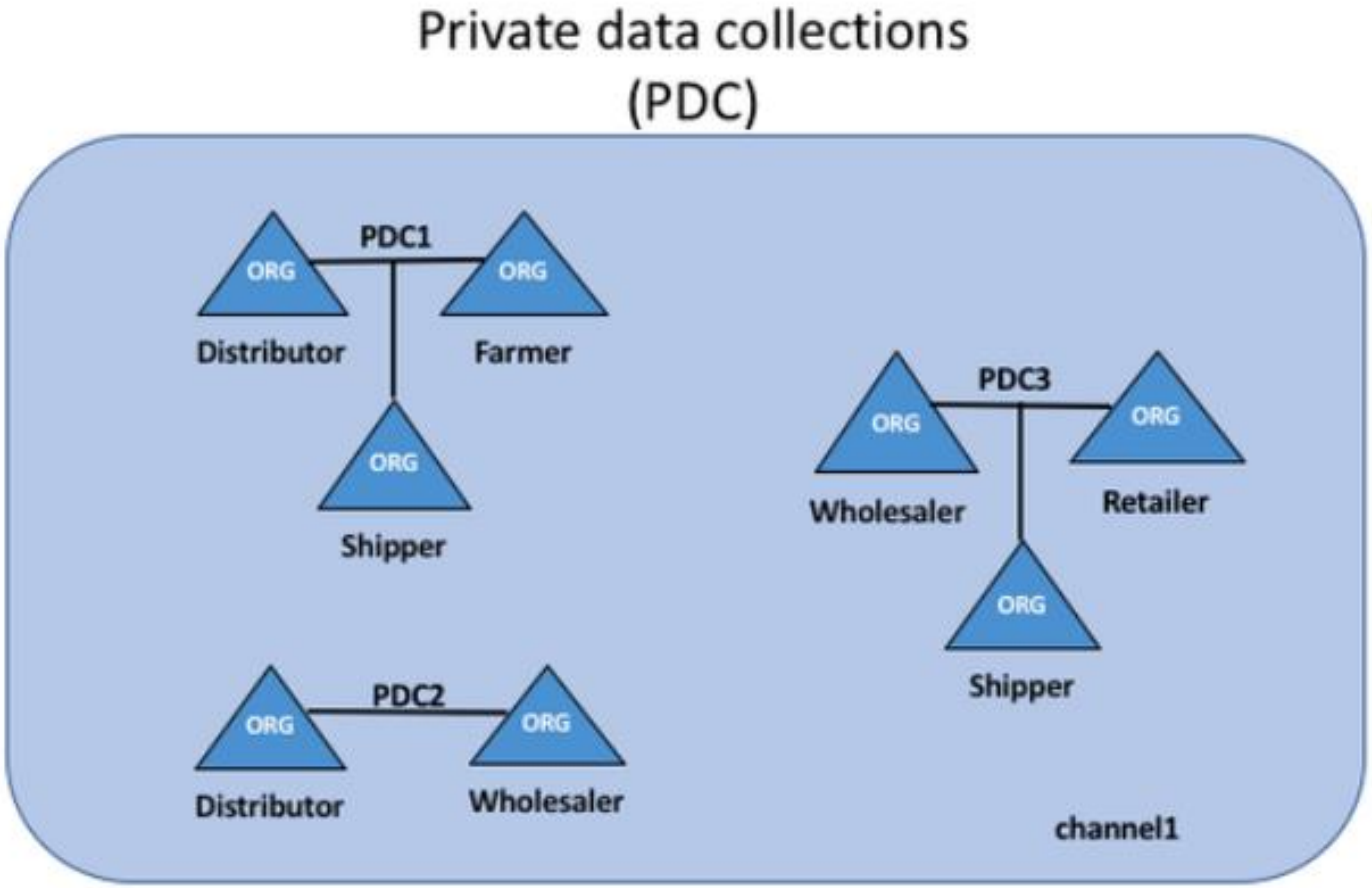


- 채널의 모든 멤버에게 트랜잭션의 비밀로 유지 되어야 할 때 ➔ Channel
- 모든 조직들 간에 트랜잭션이 공유되어야 하지만, 해당 조직의 하위 집합만 트랜잭션의 일부에 액세스 할 때 ➔ Collection (***Peer-to-peer, Don't use Orderer***)

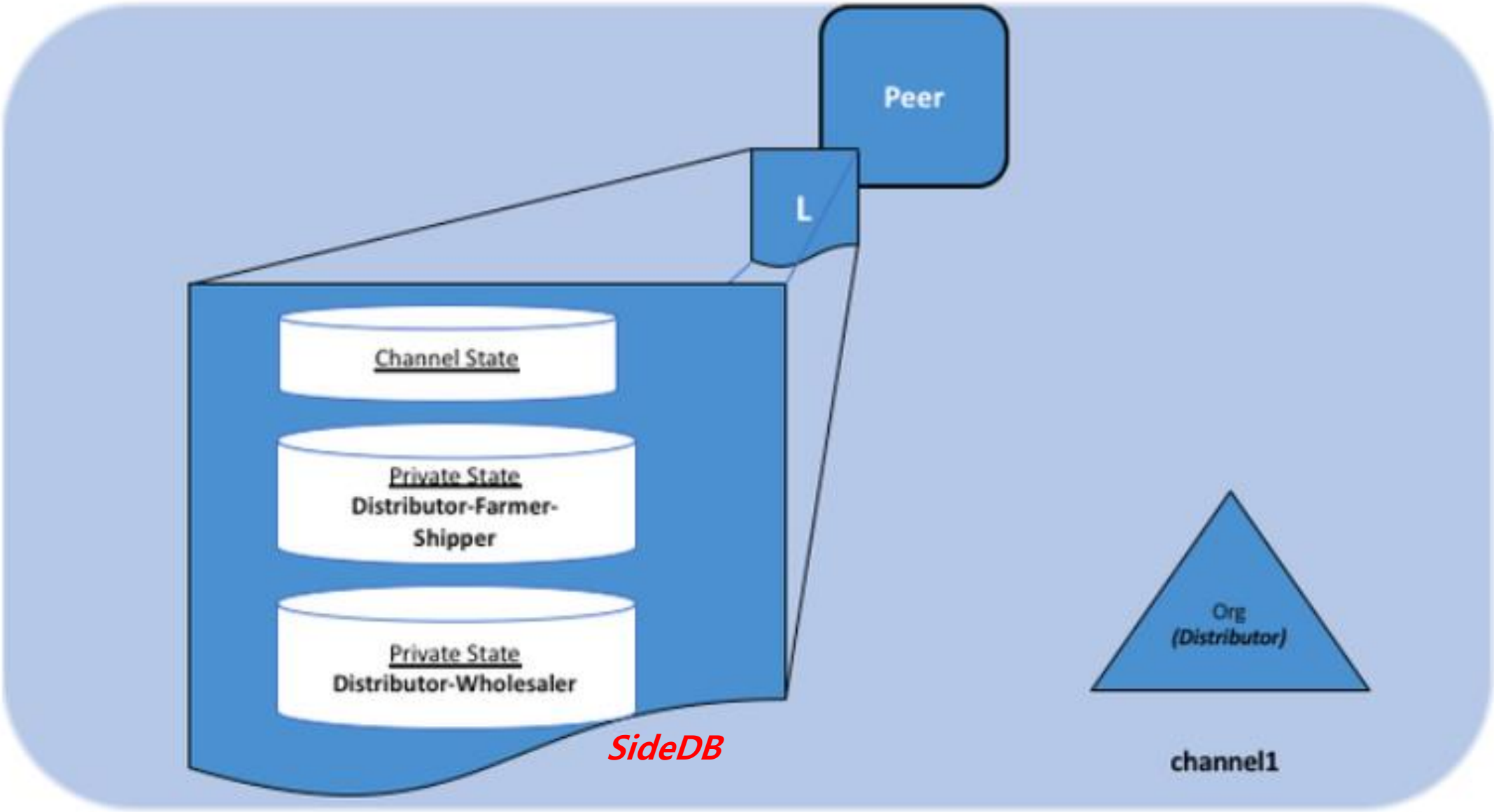
Private Data

- **A Farmer** selling his goods abroad
- **A Distributor** moving goods abroad
- **A Shipper** moving goods between parties
- **A Wholesaler** purchasing goods from distributors
- **A Retailer** purchasing goods from shippers and wholesalers

1.PDC1: **Distributor**, **Farmer** and **Shipper**
2.PDC2: **Distributor** and **Wholesaler**
3.PDC3: **Wholesaler**, **Retailer** and **Shipper**



Private Data



Private Data

1. **Build a collection definition JSON file**
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

- name
- policy
- requiredPeerCount
- maxPeerCount
- blockToLive

```
// collections_config.json

[
  {
        "name": "collectionMarbles",
        "policy": "OR('Org1MSP.member', 'Org2MSP.member')",
        "requiredPeerCount": 0,
        "maxPeerCount": 3,
        "blockToLive":1000000
  },

  {
        "name": "collectionMarblePrivateDetails",
        "policy": "OR('Org1MSP.member')",
        "requiredPeerCount": 0,
        "maxPeerCount": 3,
        "blockToLive":3
  }
]
```

Private Data

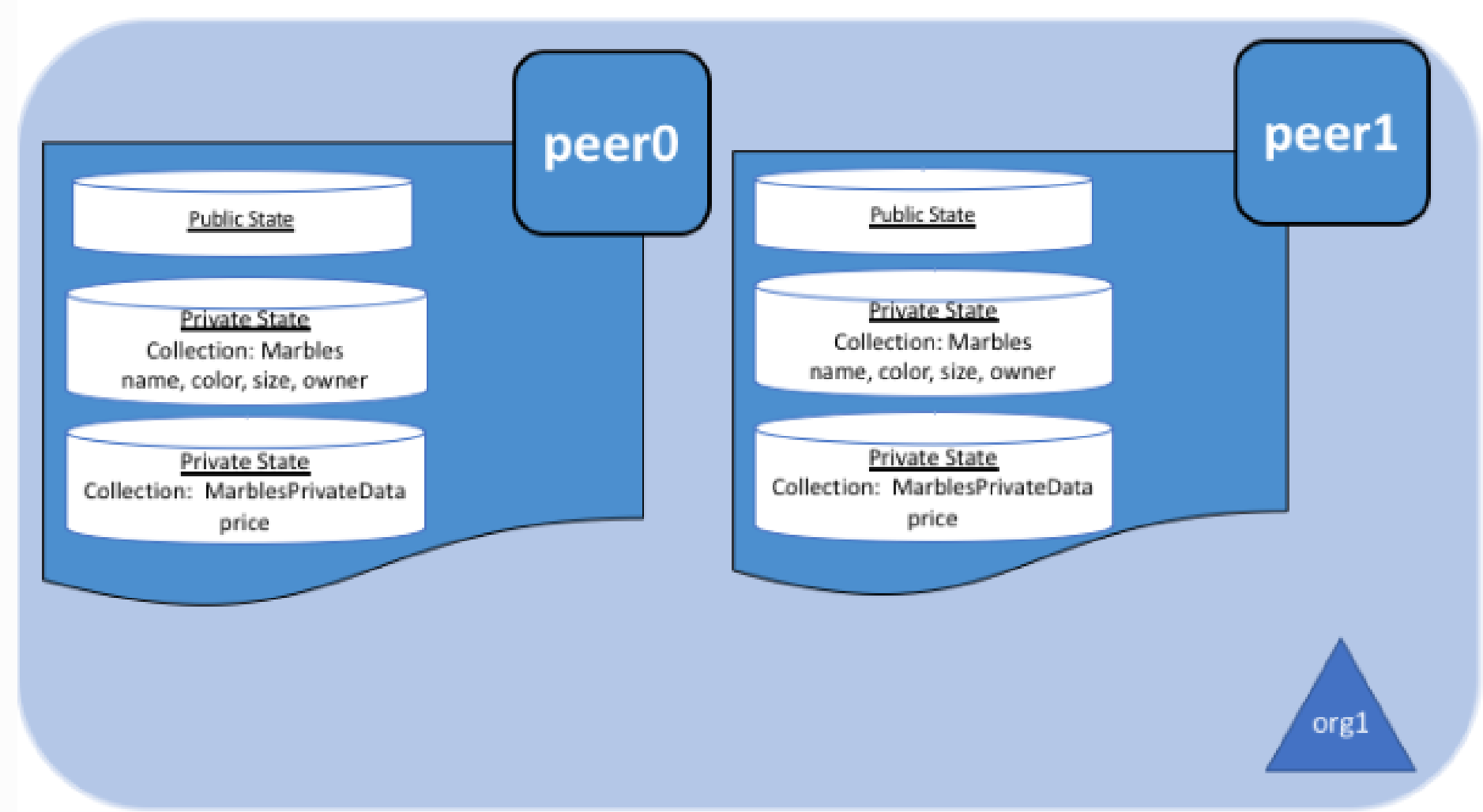
1. Build a collection definition JSON file
2. **Read and Write private data using chaincode APIs**
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
// Peers in Org1 and Org2 will have this private data in a side database
type marble struct {
  ObjectType string `json:"docType"`
  Name       string `json:"name"`
  Color      string `json:"color"`
  Size       int    `json:"size"`
  Owner      string `json:"owner"`
}

// Only peers in Org1 will have this private data in a side database
type marblePrivateDetails struct {
  ObjectType string `json:"docType"`
  Name       string `json:"name"`
  Price      int    `json:"price"`
}
```

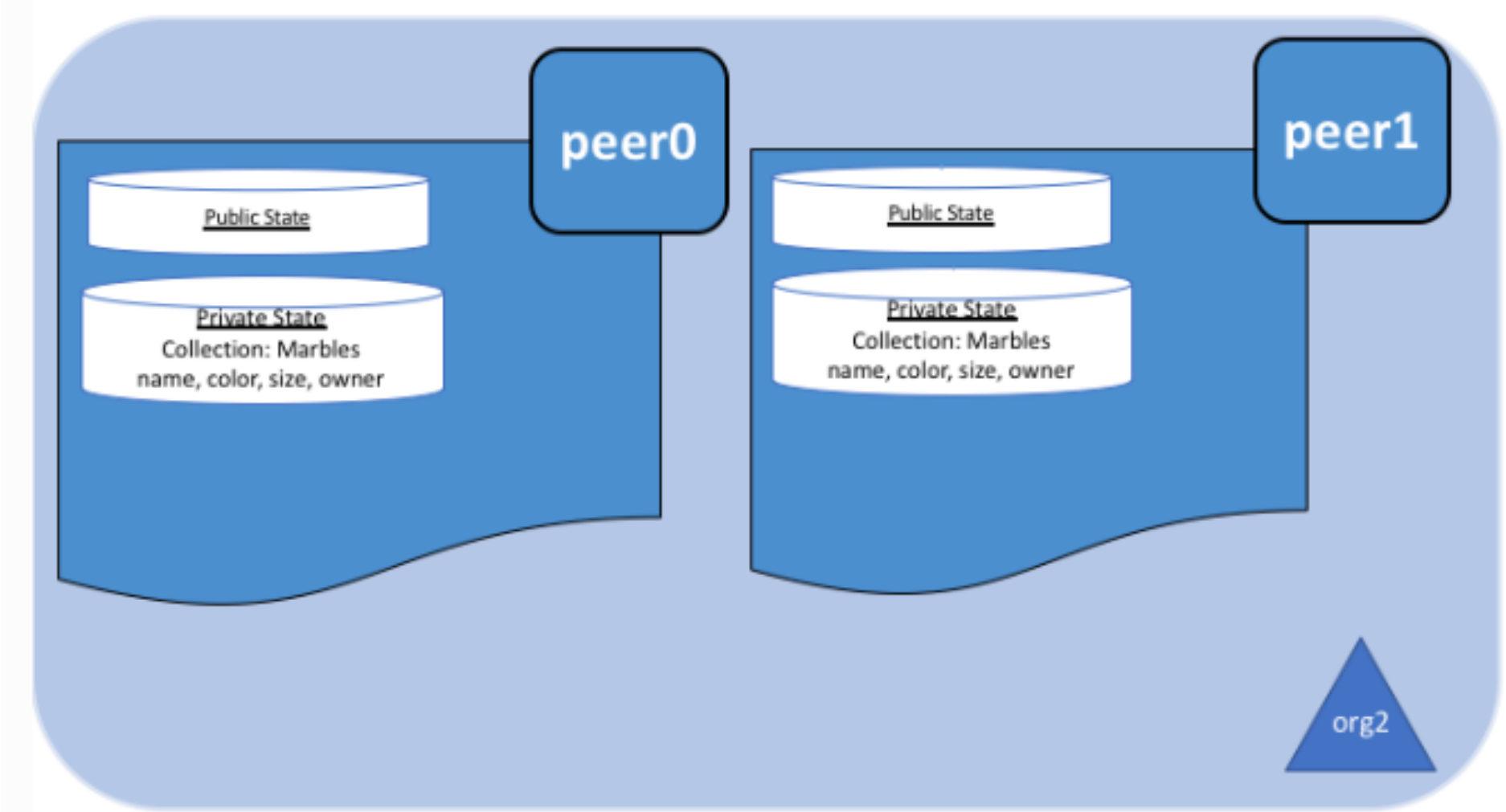
Private Data

1. Build a collection definition JSON file
2. **Read and Write private data using chaincode APIs**
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data



Private Data

1. Build a collection definition JSON file
2. **Read and Write private data using chaincode APIs**
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data



Private Data

1. Build a collection definition JSON file
2. **Read and Write private data using chaincode APIs**
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

Reading collection data

- GetPrivateData(): private data 쿼리
  - collection name
  - the data key

collections_config.json

- collectionMarbles
  - Org1, Org2의 멤버가 SideDB에 접속
- collectionMarblesPrivateDetails
  - Org1의 구성원만 SideDB에 접속

- readMarble: name, color, size and owner
- readMatrblePrivateDetails: price

Private Data

1. Build a collection definition JSON file
2. **Read and Write private data using chaincode APIs**
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

Writing private data

- collectionMarbles: name, color, size and owmer
- collectionMarblePrivateDetails: price

```
// ==== Create marble object and marshal to JSON ====
objectType := "marble"
marble := &marble{objectType, marbleName, color, size, owner}
marbleJSONasBytes, err := json.Marshal(marble)
if err != nil {
        return shim.Error(err.Error())
}

// === Save marble to state ===
err = stub.PutPrivateData("collectionMarbles", marbleName, marbleJSONasBytes)
if err != nil {
        return shim.Error(err.Error())
}

// ==== Save marble private details ====
objectType = "marblePrivateDetails"
marblePrivateDetails := &marblePrivateDetails{objectType, marbleName, price}
marblePrivateDetailsBytes, err := json.Marshal(marblePrivateDetails)
if err != nil {
        return shim.Error(err.Error())
}
err = stub.PutPrivateData("collectionMarblePrivateDetails", marbleName, marblePrivateDetailsBytes)
if err != nil {
        return shim.Error(err.Error())
}
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. **Install and instantiate chaincode with a collection**
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
cd fabric-samples/first-network
./byfn.sh down

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

- peer0.org1.example.com
- peer1.org1.example.com
- peer0.org2.example.com
- peer1.org2.example.com

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. **Install and instantiate chaincode with a collection**
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
docker exec -it cli bash
```

- ***peer0.org1.example.com***

```
peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02_private/go/
```

```
install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
```

- ***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/
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. **Install and instantiate chaincode with a collection**
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

- ***Org2 전환***

```
export CORE_PEER_LOCALMSPID=Org2MSP
export
PEER0_ORG2_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizat
ions/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/pee
rOrganizations/org2.example.com/users/Admin@org2.example.com/msp
```

- ***peer0.org2.example.com***

```
export CORE_PEER_ADDRESS=peer0.org2.example.com:7051
peer chaincode install -n marblesp -v 1.0 -p github.com/chaincode/marbles02_private/go/
```

- ***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/
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. **Install and instantiate chaincode with a collection**
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

- ***Instantiate***

```
export
ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOr
ganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exam
ple.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
```

```
[chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
[chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. **Store private data**
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
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/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
```

```
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"]}'
```

```
[chaincodeCmd] chaincodeInvokeOrQuery->INFO 001 Chaincode invoke successful. result: status:200
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. **Query the private data as an authorized peer**
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
// ============================================
// readMarble - read a marble from chaincode state
// ============================================

func (t *SimpleChaincode) readMarble(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var name, jsonResp string
    var err error
    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting name of the marble to query")
    }

    name = args[0]
    valAsbytes, err := stub.GetPrivateData("collectionMarbles", name) //get the marble from chaincode state

    if err != nil {
        jsonResp = "{₩"Error₩":₩"Failed to get state for " + name + "₩"}"
        return shim.Error(jsonResp)
    } else if valAsbytes == nil {
        jsonResp = "{₩"Error₩":₩"Marble does not exist: " + name + "₩"}"
        return shim.Error(jsonResp)
    }

    return shim.Success(valAsbytes)
}
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. **Query the private data as an authorized peer**
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

```
// ============================================
// readMarblePrivateDetails - read a marble private details from chaincode state
// ============================================

func (t *SimpleChaincode) readMarblePrivateDetails(stub shim.ChaincodeStubInterface, args []string)
pb.Response {
    var name, jsonResp string
    var err error

    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting name of the marble to query")
    }

    name = args[0]
    valAsbytes, err := stub.GetPrivateData("collectionMarblePrivateDetails", name) //get the marble private
details from chaincode state

    if err != nil {
        jsonResp = "{₩"Error₩":₩"Failed to get private details for " + name + ": " + err.Error() + "₩"}"
        return shim.Error(jsonResp)
    } else if valAsbytes == nil {
        jsonResp = "{₩"Error₩":₩"Marble private details does not exist: " + name + "₩"}"
        return shim.Error(jsonResp)
    }
    return shim.Success(valAsbytes)
}
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. **Query the private data as an authorized peer**
6. Query the private data as an unauthorized peer
7. Purge Private Data
8. Using indexes with private data

- *Org1 → marble1의 이름, 색상, 크기 및 소유자 private data*

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

```
{"color":"blue","docType":"marble","name":"marble1","owner":"tom","size":35}
```

- *Org1 → marble1의 가격 private data*

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

```
{"docType":"marblePrivateDetails","name":"marble1","price":99}
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. **Query the private data as an unauthorized peer**
7. Purge Private Data
8. Using indexes with private data

- ***Switch to a peer in Org2***

```
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
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. **Query the private data as an unauthorized peer**
7. Purge Private Data
8. Using indexes with private data

- ***Query private data Org2 is authorized to***

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

```
{"docType":"marble","name":"marble1","color":"blue","size":35,"owner":"tom"}
```

- ***Query private data Org2 is not authorized to***

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

```
{"Error":"Failed to get private details for marble1: GET_STATE failed:
transaction ID: b04adebbf165ddc90b4ab897171e1daa7d360079ac18e65fa15d84ddfebfae90:
Private data matching public hash version is not available. Public hash
version = &version.Height{BlockNum:0x6, TxNum:0x0}, Private data version =
(*version.Height)(nil)"}
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***수명을 가지는 Private data***

```
// collections_config.json

[
  {
        "name": "collectionMarbles",
        "policy": "OR('Org1MSP.member', 'Org2MSP.member')",
        "requiredPeerCount": 0,
        "maxPeerCount": 3,
        "blockToLive":1000000
  },

  {
        "name": "collectionMarblePrivateDetails",
        "policy": "OR('Org1MSP.member')",
        "requiredPeerCount": 0,
        "maxPeerCount": 3,
        "blockToLive":3
  }
]
```

3블럭까지만 저장

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Switch to a peer in Org1***

```
export CORE_PEER_ADDRESS=peer0.org1.example.com:7051
export CORE_PEER_LOCALMSPID=Org1MSP
export PEER0_ORG2_CA=/opt/gopath/src/github.com/hyperledger/₩
                     fabric/peer/crypto/peerOrganizations/org1.example.com/₩
                     peers/peer0.org1.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/org1.example.com/₩
                     users/Admin@org1.example.com/msp
```

- ***Private data logs***

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

```
[pvtdatastorage] func1 → INFO 023 Purger started: Purging expired private data till block number [0]
[pvtdatastorage] func1 → INFO 024 Purger finished
[kvledger] CommitWithPvtData → INFO 022 Channel [mychannel]: Committed block [0] with 1 transaction(s)
[kvledger] CommitWithPvtData → INFO 02e Channel [mychannel]: Committed block [1] with 1 transaction(s)
[kvledger] CommitWithPvtData → INFO 030 Channel [mychannel]: Committed block [2] with 1 transaction(s)
[kvledger] CommitWithPvtData → INFO 036 Channel [mychannel]: Committed block [3] with 1 transaction(s)
[kvledger] CommitWithPvtData → INFO 03e Channel [mychannel]: Committed block [4] with 1 transaction(s)
```

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Peer Container → query for the marble1***

```
peer chaincode query -C mychannel -n marblesp \
        -c '{"Args":["readMarblePrivateDetails","marble1"]}'
{"docType":"marblePrivateDetails","name":"marble1","price":99}
```

- ***Create marble2***

```
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"]}'
```

- ***Private data logs***

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

→ 블록 높이 1 증가

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Peer Container → query for the marble1 again***

```
peer chaincode query -C mychannel -n marblesp \
        -c '{"Args":["readMarblePrivateDetails","marble1"]}'
{"docType":"marblePrivateDetails","name":"marble1","price":99}
```

- ***Transfer marble2 to "joe" ➔ 두번째 블록 추가***

```
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":["transferMatble","marble2","joe"]}'
```

- ***Private data logs***

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

→ 블록 높이 1 증가

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Peer Container → query for the marble1 again***

```
peer chaincode query -C mychannel -n marblesp \
        -c '{"Args":["readMarblePrivateDetails","marble1"]}'
{"docType":"marblePrivateDetails","name":"marble1","price":99}
```

- ***Transfer marble2 to "tom" → 세번째 블록 추가***

```
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":["transferMatble","marble2", "tom"]}'
```

- ***Private data logs***

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

→ 블록 높이 1 증가

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Peer Container → query for the marble1 again***

```
peer chaincode query -C mychannel -n marblesp \
        -c '{"Args":["readMarblePrivateDetails","marble1"]}'
{"docType":"marblePrivateDetails","name":"marble1","price":99}
```

- ***Transfer marble2 to "tom" → 네번째 블록 추가 → price private data 삭제***

```
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":["transferMatble","marble2","jerry"]}'
```

- ***Private data logs***

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

→ 블록 높이 1 증가

Private Data

1. Build a collection definition JSON file
2. Read and Write private data using chaincode APIs
3. Install and instantiate chaincode with a collection
4. Store private data
5. Query the private data as an authorized peer
6. Query the private data as an unauthorized peer
7. **Purge Private Data**
8. Using indexes with private data

- ***Peer Container → query for the marble1 again***

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

```
Error: endorsement failure during query. response: status:500
message:"{\"Error\":\"Marble private details does not exist: marble1\"}"
```