Hyperledger Fabric

v2.4

with Java

김성부

목차

개발환경

- OS
  - Linux
    - ubuntu 20.04
  - Windows
    - Windows 10
    - WSL2 Unbuntu 20.04
    - Docker Desktop /w 'Use the WSL 2 based engine'
- 필요 프로그램
  - git 2.25.1
  - curl 7.68.0
  - docker 29.10.7, docker-compose 1.25.0
  - node.js 16.14.0, npm 8.3.1
  - java(openjdk) 11.0.13, gradle 7.4
  - go 1.17.7
  - Jq 1.6
- Hyperledger fabric
  - fabric 2.4.2 (최신 버전 2.4.3도 호환될 것으로 예상)
  - fabric-ca 1.5.2

용어

용 어	설 명
신원확인(Identity)	네트워크 참가자의 인증정보 (X.509: PrivateKey + Certificate)
원장(Ledger)	블록의 트랜잭션(거래) 정보가 실제로 저장되는 공간
트랜잭션(Transaction)	스마트 컨트랙트인 체인코드의 실행
체인코드(Chaincode)	하이퍼레저 패브릭의 스마트 컨트랙트 (Go, Node.js, Java)
채널(Channel)	트랜잭션의 접근 권한을 조직별로 설정하고 관리
조직(Organization)	네트워크 구성 단위 조직 별로 노드 관리 및 권한 부여, 보증 정책 등을 수행
피어	트랜잭션 제안 및 응답을 처리, 원장과 체인코드를 관리하고 저장
오더링 서비스	트랜잭션 정렬 및 블록 생성 합의 알고리즘: raft
MSP	인증 서비스 (Membership Service Provider)
CA(Certification Authority)	MSP에서 암호화 인증을 위해 필요한 인증기관 공개키 인증서 및 이에 대응하는 개인 키를 발급

용어 – Identity classification

분 류	설 명
client	네트워크에서 거래(트랜잭션)
admin	관리 업무 처리(채널에 peer 참여(join), 채널설정 수정 트랜잭션의 서명 등)
peer	트랜잭션 보증(endorese) 및 커밋(commit)
orderer	트랜잭션 정렬(order) 및 블록 생성

용어 – 원장(Ledger)

- 블록체인
  - HASH: SHA-256
  - 전자서명
    - 256 : ecdsa – prime256v1 (default)
    - 384 : ecdsa - secp384r1
    - 521 : ecdsa - secp521r1
  - 합의 알고리즘: raft – 링크 (CFT, crash fault tolerant)(in etcd)
  - 스마트 계약: Go, Node.js, Java
  - 통신: gRPC
  - 운영환경: container (docker, podman, k8s 등)
- 월드 스테이트(World State)
  - levelDB : 단순 key/value (default)
  - CouchDB : value를 json으로 저장하여 selector로 쿼리 가능

용어 – Peer

가장 기본적인 네트워크 구성 요소로, 블록체인 네트워크를 유지하고 **트랜잭션의 제안 및 응답을 처리하며, 원장과 체인코드를 관리하고 저장**하는 역할을 수행한다. 피어 노드는 역할에 따라 다음과 같은 4가지로 세분화될 수 있다.

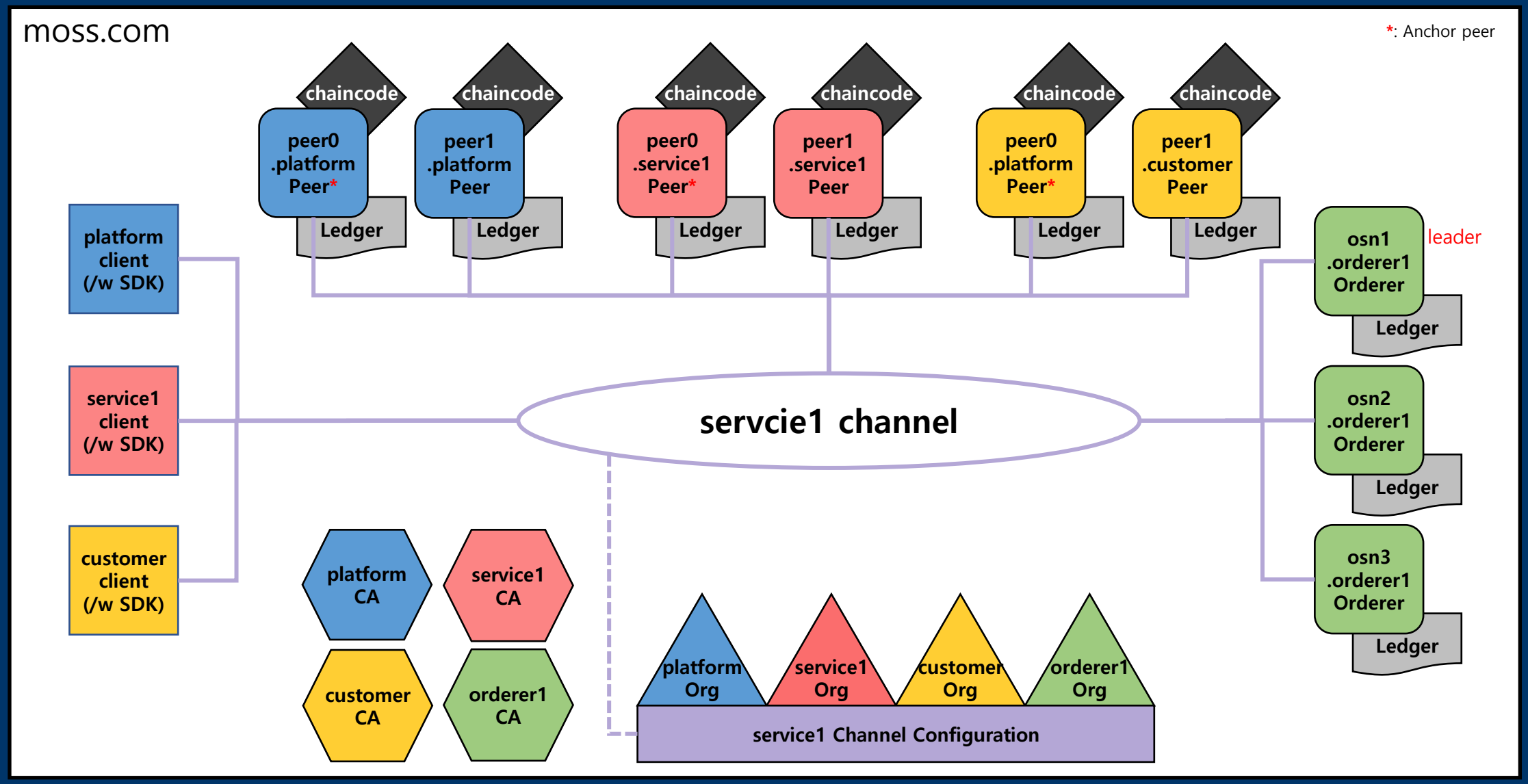
피어 노드	설 명
엔도싱(Endorsing)	보증 정책(Endorsing Policy)에 따라 요청된 트랜잭션을 먼저 실행해보는 검토 역할을 수행한 후 트랜잭션에 트랜잭션 보증 사인을 첨부하는 역할을 한다.
커미팅(Committing)	엔도싱 피어 노드가 실행한 트랜잭션 결과를 검증하고 문제가 없으면 트랜잭션을 확정하고 그 내용을 블록체인에 업데이트하는 역할을 한다.
앵커(Anchor)	채널 내에서 대표 역할을 수행하는 피어 노드다.
리더(Leader)	조직에서 모든 피어 노드를 대표한다. Orderer로 부터 직접 블록을 전파받는다.

네트워크 & 채널

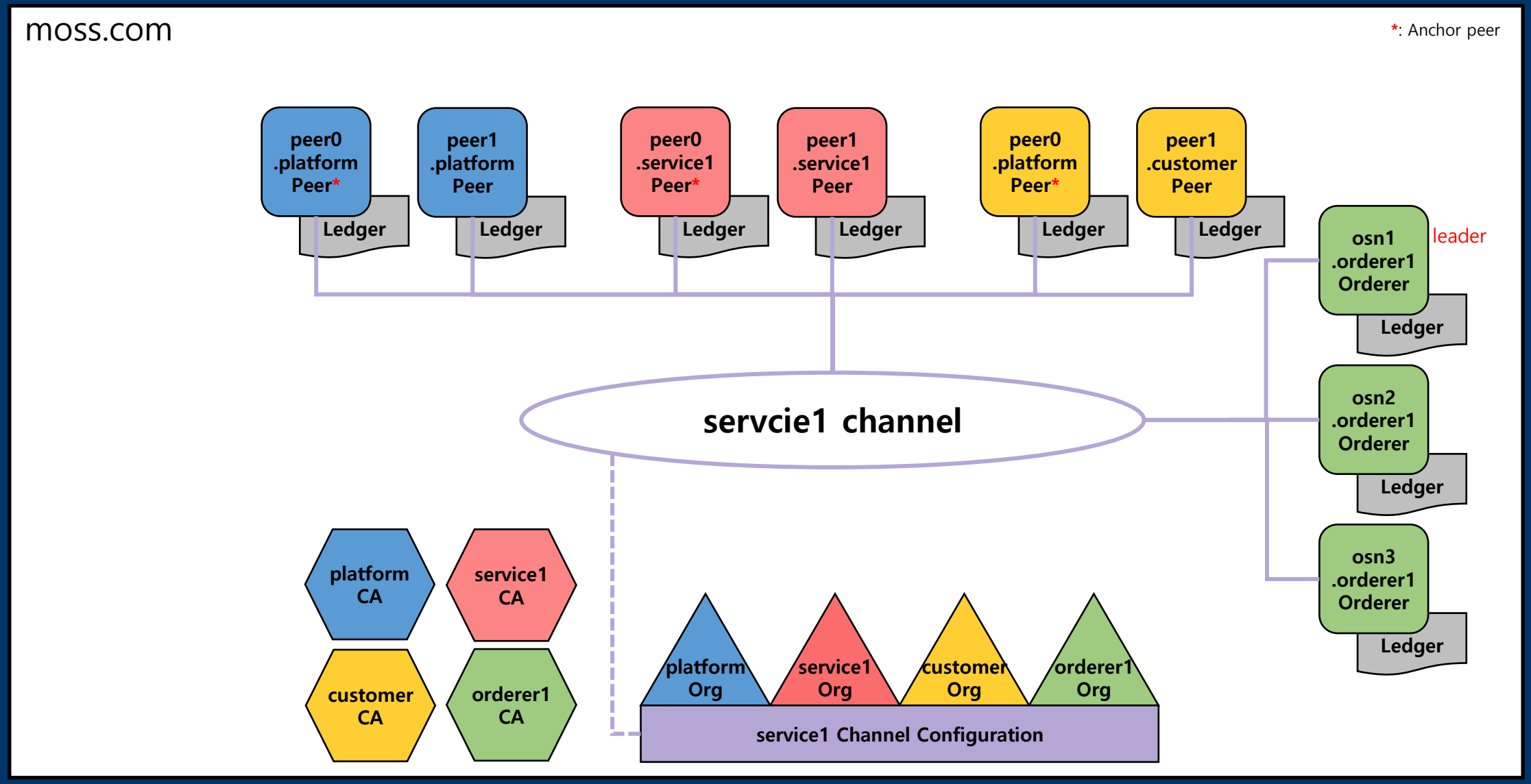
임의의 '서비스1'를 위한 채널 구성

- 채널: service1
- Peer 조직
  - platform : peer 2개(peer0, peer1)
  - customer : peer 2개(peer0, peer1)
  - service1 : peer 2개(peer0, peer1)
- Orderer 조직
  - orderer1 : orderer 3개 (osn1, osn2, osn3) [cluster 구성]

네트워크 & 채널

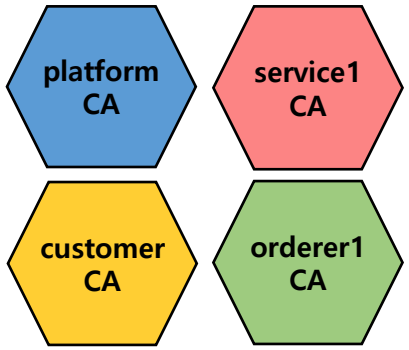

moss.com
*: Anchor peer
chaincode
peer0 .platform Peer*
peer1 .platform Peer
peer0 .service1 Peer*
peer1 .service1 Peer
peer0 .platform Peer*
peer1 .customer Peer
Ledger
platform client (/w SDK)
service1 client (/w SDK)
customer client (/w SDK)
servcie1 channel
osn1 .orderer1 Orderer
leader
osn2 .orderer1 Orderer
osn3 .orderer1 Orderer
platform CA
service1 CA
customer CA
orderer1 CA
platform Org
service1 Org
customer Org
orderer1 Org
service1 Channel Configuration

네트워크 & 채널


moss.com
*: Anchor peer
peer0 .platform Peer*
peer1 .platform Peer
peer0 .service1 Peer*
peer1 .service1 Peer
peer0 .platform Peer*
peer1 .customer Peer
Ledger
osn1 .orderer1 Orderer
leader
osn2 .orderer1 Orderer
osn3 .orderer1 Orderer
servcie1 channel
platform CA
service1 CA
customer CA
orderer1 CA
platform Org
service1 Org
customer Org
orderer1 Org
service1 Channel Configuration

네트워크 & 채널 – 1. 조직별 CA 구성

moss.com



네트워크 & 채널 – 1. 조직별 CA 구성

organizations/fabric-ca/platform/fabric-ca-server-config.yaml

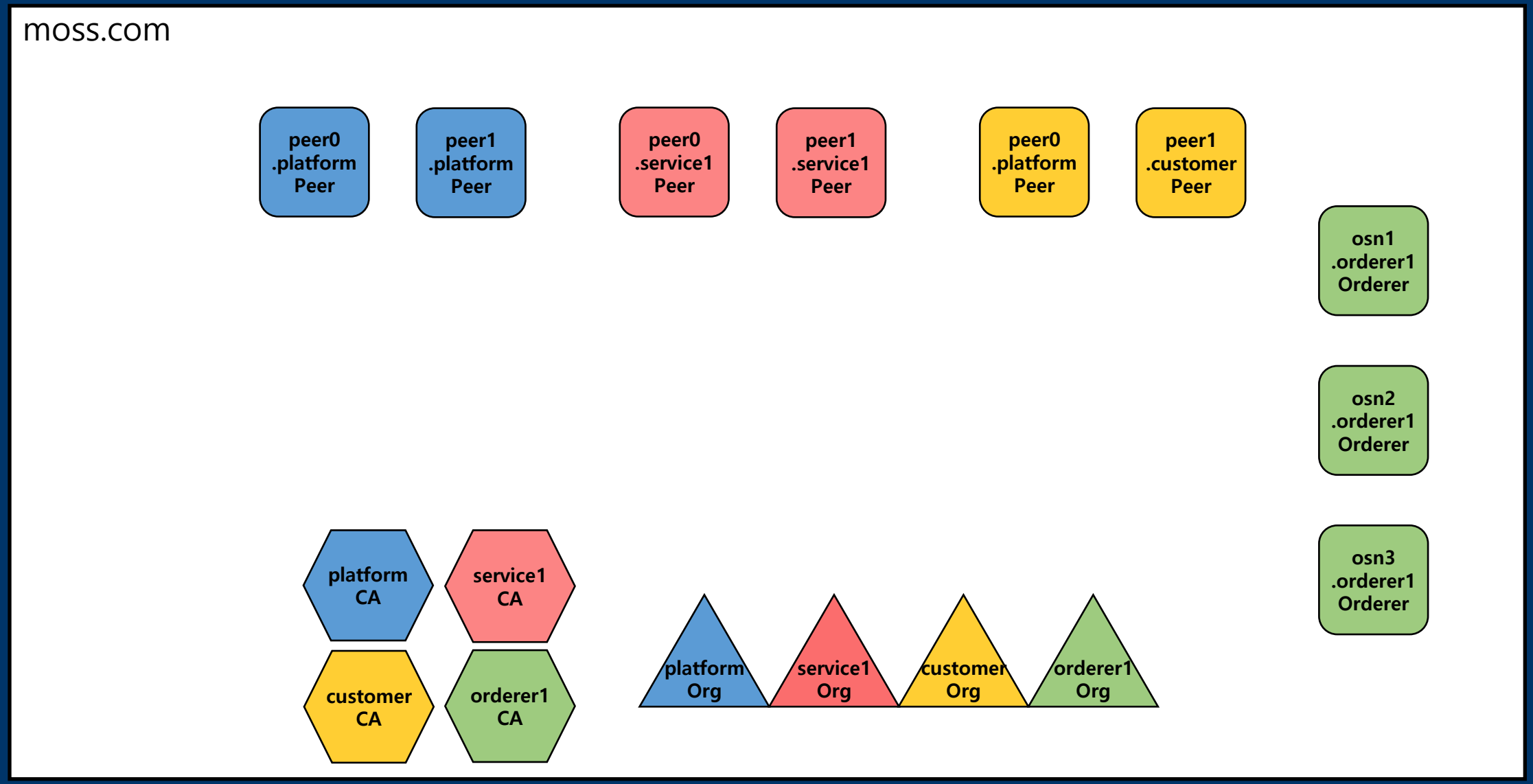
```yaml
version: 1.2.0
port: 8054
ca:
  name: ca.platform.moss.com
registry:
  # Contains identity information which is used when LDAP is disabled
  identities:
    - name: padmin
      pass: padminpw
      type: client
db:
  type: sqlite3 # sqlite3, mysql, postgre
# Certificate Signing Request
csr:
  names:
    - C: KR
      ST: "Gyeonggi-Do"
      L: "Seongnam-si"
      O: MOSS Corporation
      OU:
  hosts:
    - localhost
    - platform.moss.com
  key:
    algo: ecdsa
    size: 256
# BlockChain Crypto Service Provider
bccsp:
  default: SW
  sw:
    hash: SHA2
    security: 256
…
```

네트워크 & 채널 – 1. 조직별 CA 구성

CA Containers

```
$ docker container ls --format "table {{.ID}}\t{{.Names}}\t{{.Ports}}\t{{.Image}}"
CONTAINER ID   NAMES         PORTS                              IMAGE
853bc7a58e3b   ca_orderer1   0.0.0.0:7054->7054/tcp …           hyperledger/fabric-ca:1.5.2
e33d322846fe   ca_platform   0.0.0.0:8054->8054/tcp …           hyperledger/fabric-ca:1.5.2
8222fb771531   ca_customer   0.0.0.0:9054->9054/tcp …           hyperledger/fabric-ca:1.5.2
62e244741023   ca_service1   0.0.0.0:10054->10054/tcp …         hyperledger/fabric-ca:1.5.2
```

네트워크 & 채널 – 2. 조직별 Identity 작업



네트워크 & 채널 – 2. 조직별 Identity 작업

fabric-ca-client-config.yaml – Identity 설정 일부

```yaml
# URL of the Fabric-ca-server (default: http://localhost:7054)
url: https://localhost:8054
# Name of the CA to connect to within the fabric-ca server
caname: ca.platform.moss.com
id:
  name: peer0
  secret: peer0pw
  type: peer
# Certificate Signing Request
csr:
  keyrequest:
    algo: ecdsa
    size: 256
  names:
    - C: KR
      ST: "Gyeonggi-Do"
      L: "Seongnam-si"
      O: MOSS Corporation
      OU: Platform
  hosts:
    - localhost
    - peer0.platform.moss.com
...
```

네트워크 & 채널 – 2. 조직별 Identity 작업

$FABRIC_CFG_PATH/core.yaml – peer 설정 파일 일부 (1/2)

```yaml
peer:
  tls:
    enabled: true
    cert.file: /etc/hyperledger/fabric/tls/server.crt
    key.file: /etc/hyperledger/fabric/tls/server.key
    rootcert.files: /etc/hyperledger/fabric/tls/ca.crt
  profile.enbaled: false # it should be disabled (eg enabled: false)
  id: peer0.platform.moss.com
  listenAddress: 0.0.0.0:8060
  address: peer0.platform.moss.com:8060
  chaincodeAddress: peer0.platform.moss.com:8070
  chaincodeListenAddress: 0.0.0.0:8070

  gateway.enabled: true

  gossip:
    bootstrap: peer0.platform.moss.com:8060
    externalEndpoint: peer0.platform.moss.com:8060
    orgLeader: true

  localMspId: platformrg
  mspConfigPath: /etc/hyperledger/fabric/msp
  localMspType: bccsp
  # BCCSP (Blockchain crypto provider): Select which crypto implementation or library to use
  BCCSP:
    Default: SW
    # Settings for the SW crypto provider (i.e. when DEFAULT: SW)
    SW:
      Hash: SHA2
      Security: 256
```

네트워크 & 채널 – 2. 조직별 Identity 작업

$FABRIC_CFG_PATH/core.yaml – peer 설정 파일 일부 (2/2)

```yaml
vm:
  endpoint: unix:///host/var/run/docker.sock
  docker:
    hostconfig:
      NetworkMode: fabric_moss # docker-compose 네트워크 이름(sets the networking mode for the container)

chaincode:
  externalBuilders:
    - name: ccaas_builder
      path: /opt/hyperledger/ccaas_builder
      propagateEnvironment:
        - CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG # {"peername":"peer0platform"}
  installTimeout: 300s
  startuptimeout: 300s
  executetimeout: 300s
  logging:
    level: info
    shim: warning
ledger:
  state:
    stateDatabase: goleveldb # goleveldb, CouchDB
    totalQueryLimit: 100000

operations:
  listenAddress: peer0.platform.moss.com:8060

merics:
  provider: prometheus  # metrics provider is one of statsd, prometheus, or disabled
```

네트워크 & 채널 – 2. 조직별 Identity 작업

$FABRIC_CFG_PATH/orderer.yaml – osn 설정 파일 일부 (1/2)

```yaml
General:
  ListenAddress: 0.0.0.0
  ListenPort: 7061
  LocalMSPID: orderer1MSP
  LocalMSPDir: /var/hyperledger/orderer/msp

  TLS:
    Enabled: true
    PrivateKey: /var/hyperledger/orderer/tls/server.key
    Certificate: /var/hyperledger/orderer/tls/server.crt
    RootCAs:
      - /var/hyperledger/orderer/tls/ca.crt
  Cluster:
    ClientCertificate: /var/hyperledger/orderer/tls/server.crt
    ClientPrivateKey: /var/hyperledger/orderer/tls/server.key
    RootCAs:
      - /var/hyperledger/orderer/tls/ca.crt

  BootstrapMethod: none # allows an orderer to start without a system channel configuration
```

네트워크 & 채널 – 2. 조직별 Identity 작업

$FABRIC_CFG_PATH/orderer.yaml – osn 설정 파일 일부 (2/2)

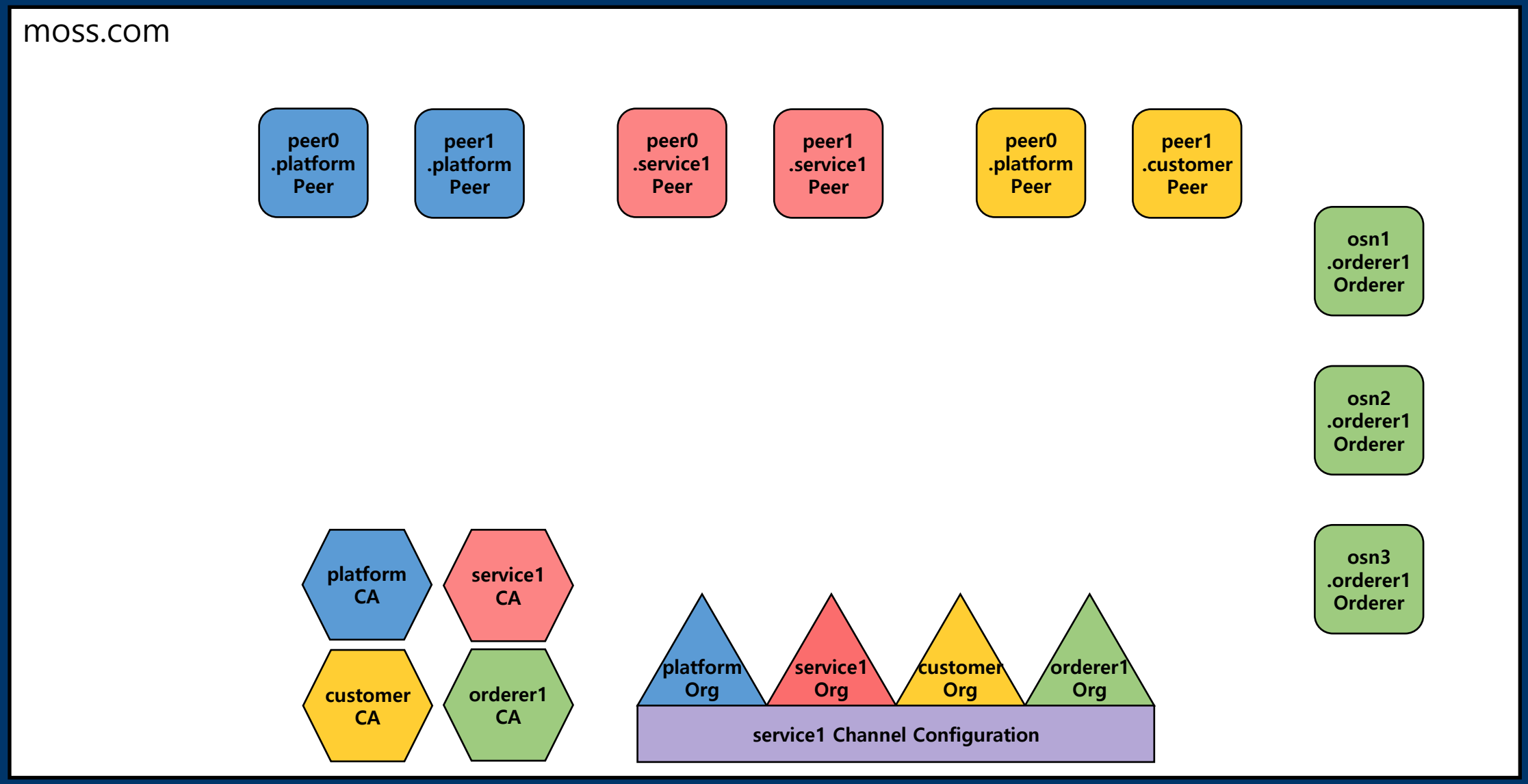
```yaml
ChannelParticipation:
  Enabled: true

Admin:
  ListenAddress: 0.0.0.0:7071
  TLS:
    Enabled: true
    Certificate: /var/hyperledger/orderer/tls/server.crt
    PrivateKey: /var/hyperledger/orderer/tls/server.key
    RootCAs: ["/var/hyperledger/orderer/tls/ca.crt"]
    ClientRootCAs: ["/var/hyperledger/orderer/tls/ca.crt"]
Operations:
  ListenAddress: osn1.orderer1.moss.com:7081

Metrics:
  # The metrics provider is one of statsd, prometheus, or disabled
  Provider: prometheus
```

네트워크 & 채널 – 2. 조직별 Identity 작업

osn, peer, ca Containers

```
$ docker container ls --format "table {{.ID}}\t{{.Names}}\t{{.Ports}}\t{{.Image}}"
CONTAINER ID   NAMES                      PORTS                          IMAGE
3db00718f0ee   osn1.orderer1.moss.com     0.0.0.0:7061->7061/tcp …       hyperledger/fabric-orderer:2.4.2
4b6cc45d5193   osn2.orderer1.moss.com     0.0.0.0:7062->7062/tcp …       hyperledger/fabric-orderer:2.4.2
9b544c3ce7e6   osn3.orderer1.moss.com     0.0.0.0:7063->7063/tcp …       hyperledger/fabric-orderer:2.4.2
4362a3580c1a   peer0.platform.moss.com    0.0.0.0:8060->8060/tcp …       hyperledger/fabric-peer:2.4.2
2ee1cc06721d   peer1.platform.moss.com    0.0.0.0:8061->8061/tcp …       hyperledger/fabric-peer:2.4.2
33441a750625   peer0.customer.moss.com    0.0.0.0:9060->9060/tcp …       hyperledger/fabric-peer:2.4.2
0172e2661e6b   peer1.customer.moss.com    0.0.0.0:9061->9061/tcp …       hyperledger/fabric-peer:2.4.2
3ab8cafb2e1d   peer0.service1.moss.com    0.0.0.0:10060->10060/tcp …     hyperledger/fabric-peer:2.4.2
fcdf87cc9b6e   peer1.service1.moss.com    0.0.0.0:10061->10061/tcp …     hyperledger/fabric-peer:2.4.2
853bc7a58e3b   ca_orderer1                0.0.0.0:7054->7054/tcp …       hyperledger/fabric-ca:1.5.2
e33d322846fe   ca_platform                0.0.0.0:8054->8054/tcp …       hyperledger/fabric-ca:1.5.2
8222fb771531   ca_customer                0.0.0.0:9054->9054/tcp …       hyperledger/fabric-ca:1.5.2
62e244741023   ca_service1                0.0.0.0:10054->10054/tcp …     hyperledger/fabric-ca:1.5.2
```

네트워크 & 채널 – 3. 채널 설정 작성



네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Orderer : Orderer Organization 공통 설정 (1/2)

```
Orderer: &OrdererDefaults
    # 합의 알고리즘: v2.4 기준 공식 지원 합의 알고리즘은 etcdraft 1개
    OrdererType: etcdraft
    # client와 peer가 접속할 수 있는 orderer 주소 목록
    Addresses:
        - osn1.orderer1.moss.com:7061
        - osn2.orderer1.moss.com:7062
        - osn3.orderer1.moss.com:7063

    EtcdRaft:
        # orderer간 통신에 사용하는 주소 및 TLS 정보
        Consenters:
        - Host: osn1.orderer1.moss.com
          Port: 7061
          ClientTLSCert: ../organizations/…/osn1.orderer1.moss.com/tls/server.crt
          ServerTLSCert: ../organizations/…/osn1.orderer1.moss.com/tls/server.crt
        - Host: osn2.orderer1.moss.com
          Port: 7062
          ClientTLSCert: ../organizations/…/osn2.orderer1.moss.com/tls/server.crt
          ServerTLSCert: ../organizations/…/osn2.orderer1.moss.com/tls/server.crt
        - Host: osn3.orderer1.moss.com
          Port: 7063
          ClientTLSCert: ../organizations/…/osn3.orderer1.moss.com/tls/server.crt
          ServerTLSCert: ../organizations/…/osn3.orderer1.moss.com/tls/server.crt
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Orderer : Orderer Organization 공통 설정 (2/2)

```
Orderer: &OrdererDefaults
    # 성능과 연관된 가장 중요한 설정
    # BatchTimeout, MaxMessageCount, PrferredMaxBytes 중 하나만 충족되면 Orderer는 블록을 생성한다.

    # Batch Timeout: The amount of time to wait before creating a batch
    # 첫번째 트랜잭션 도착 후, 블록 생성까지 대기시간
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        # 블럭당 최대 메시지 개수
        MaxMessageCount: 500

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        # 블록 최대 크기: 블록 최대 크기를 초과하는 트랙잭션은 항상 Reject
        AbsoluteMaxBytes: 10 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        # 선호 블록 크기: 트랜잭션을 추가 후 지정된 값을 초과하면 블록을 생성한다.
        PreferredMaxBytes: 2 MB
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Orderer : Orderer Organization 공통 설정 (3/3)

```
Orderer: &OrdererDefaults
    # Organizations is the list of orgs which are defined as participants on
    # the orderer side of the network
    Organizations:

    # Policies , /Channel/Orderer/<PolicyName>
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        # BlockValidation specifies what signatures must be included in the block
        # from the orderer for the peer to validate it.
        BlockValidation:
            Type: ImplicitMeta
            Rule: "ANY Writers"
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Application : Peer Organization 공통 설정

```
# The values to encode into a config transaction or genesis block for application related parameters
Application: &ApplicationDefaults
    # Organizations is the list of orgs which are defined as participants on
    # the application side of the network
    Organizations:

    # Policies , /Channel/Application/<PolicyName>
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        LifecycleEndorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"
        Endorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"

    Capabilities:
        <<: *ApplicationCapabilities
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – platform Peer 조직 설정

```
Organizations:
- &Platform
    Name: platformMSP

    ID: platformMSP

    MSPDir: ../organizations/peerOrganizations/platform.moss.com/msp

    # Policies , /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('platformMSP.admin', 'platformMSP.peer', 'platformMSP.client')"
        Writers:
            Type: Signature
            Rule: "OR('platformMSP.admin', 'platformMSP.client')"
        Admins:
            Type: Signature
            Rule: "OR('platformMSP.admin')"
        Endorsement:
            Type: Signature
            Rule: "OR('platformMSP.peer')"
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – orderer1 Orderer조직 설정

```
Organizations:
  - &Orderer1
    Name: Orderer1
    ID: orderer1MSP
    MSPDir: ../organizations/ordererOrganizations/orderer1.moss.com/msp

    # Policies, /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
      Readers:
        Type: Signature
        Rule: "OR('orderer1MSP.member')"
      Writers:
        Type: Signature
        Rule: "OR('orderer1MSP.member')"
      Admins:
        Type: Signature
        Rule: "OR('orderer1MSP.admin')"

    OrdererEndpoints:
      - osn1.orderer1.moss.com:7061
      - osn2.orderer1.moss.com:7062
      - osn3.orderer1.moss.com:7063
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Channel 공통 설정

```
Channel: &ChannelDefaults
    # Policies , /Channel/<PolicyName>
    Policies:
        # Who may invoke the 'Deliver' API
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        # Who may invoke the 'Broadcast' API
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        # By default, who may modify elements at this config level
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"

    # Capabilities describes the channel level capabilities, see the
    # dedicated Capabilities section elsewhere in this file for a full
    # description
    Capabilities:
        <<: *ChannelCapabilities
```

네트워크 & 채널 – 3. 채널 설정 작성

채널 설정 – Capabilities (버전 호환성) 설정

```
Capabilities:
    Channel: &ChannelCapabilities
        V2_0: true
    Orderer: &OrdererCapabilities
        V2_0: true
    Application: &ApplicationCapabilities
        V2_0: true
```

네트워크 & 채널 – 3. 채널 설정 작성

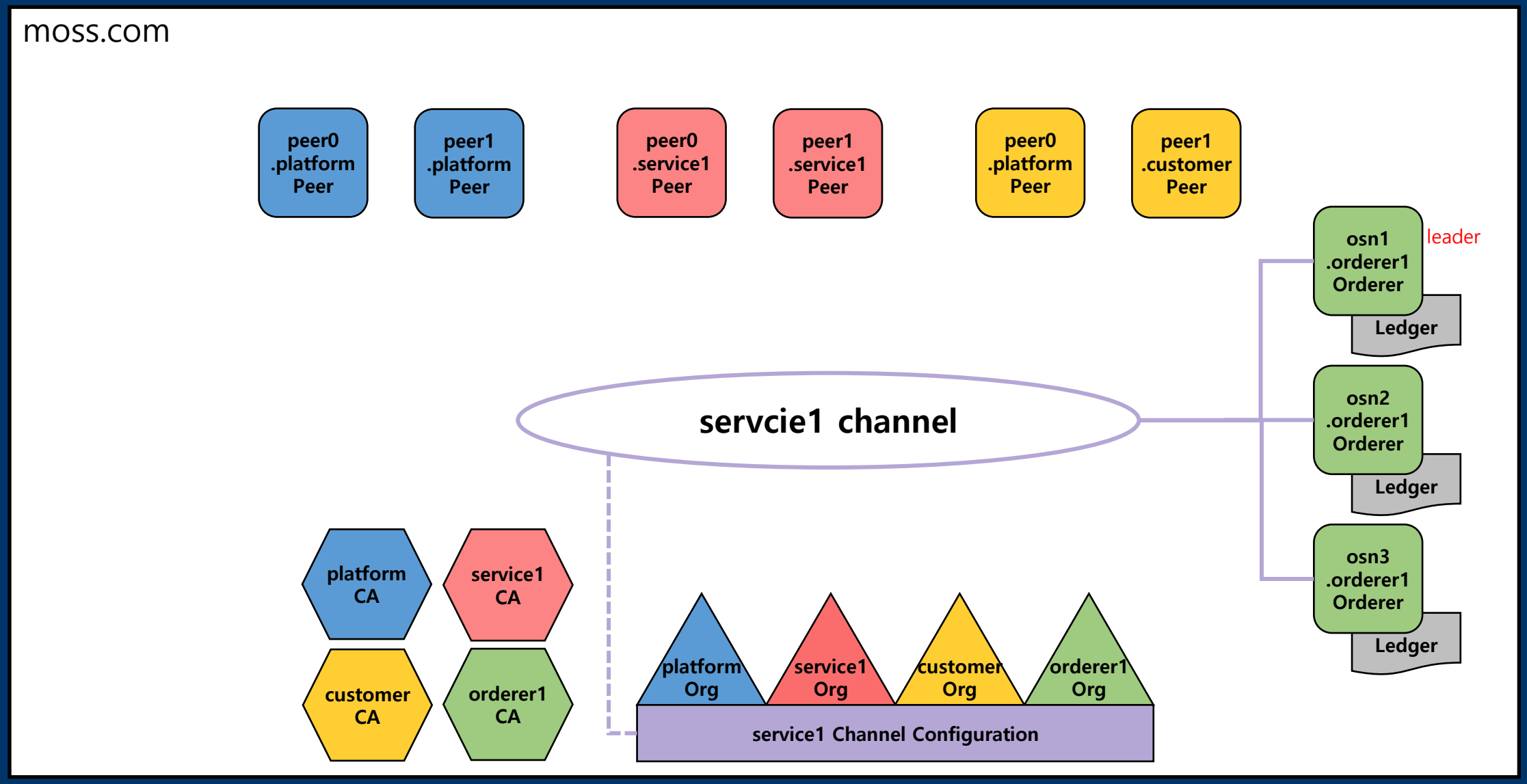
채널 설정 – Profile

```
Profiles:
    Service1ApplicationGenesis:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *Orderer1
            Capabilities: *OrdererCapabilities
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Platform
                - *Customer
                - *service1
            Capabilities: *ApplicationCapabilities
```

채널 genesis block 생성 (Profile 지정)

```
export FABRIC_CFG_PATH=${PWD}/configtx
mkdir -p channel-artifacts
configtxgen -profile Service1ApplicationGenesis -outputBlock ./channel-artifacts/service1.block -channelID service1
```

네트워크 & 채널 – 4. OSN Join, 채널 생성



네트워크 & 채널 – 4. OSN Join, 채널 생성

OSN(Ordering Service Nod) Join

- 채널이 존재하지 않으면 채널 생성 후 참여
- 블록체인이 생성되는 시점

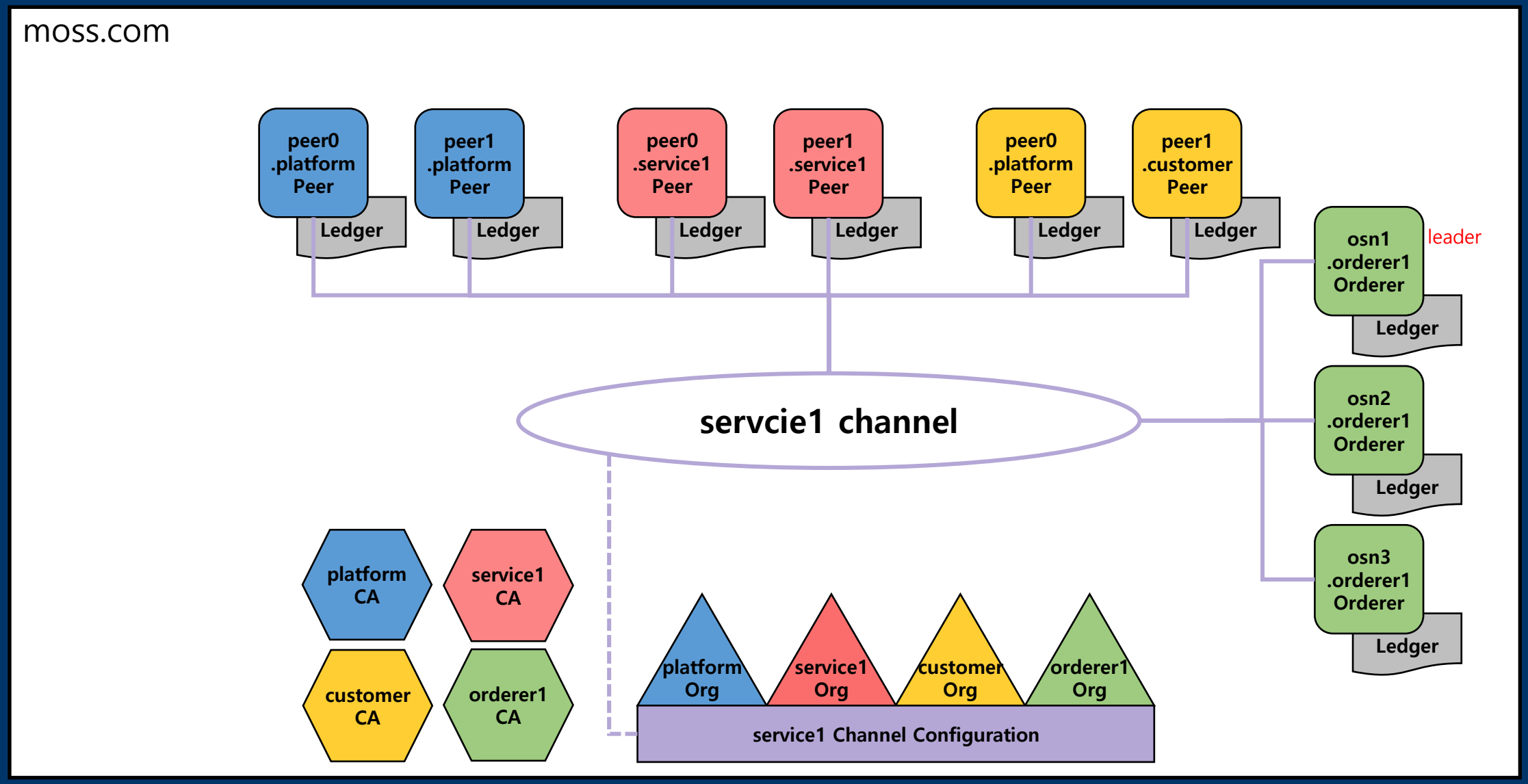
```
osnadmin channel join --channelID service1 --config-block ./channel-artifacts/service1.block -o localhost:7071 …
osnadmin channel join --channelID service1 --config-block ./channel-artifacts/service1.block -o localhost:7072 …
osnadmin channel join --channelID service1 --config-block ./channel-artifacts/service1.block -o localhost:7073 …
```

네트워크 & 채널 – 4. OSN Join, 채널 생성

합의 알고리즘(raft)에 따라 리더 선정

```
[orderer.common.cluster] updateStubInMapping -> Allocating a new stub for node 2 with endpoint of osn2.orderer1.moss.com:7062 for channel servic
[orderer.common.cluster] updateStubInMapping -> Deactivating node 2 in channel service1 with endpoint of osn2.orderer1.moss.com:7062 due to TLS
[orderer.common.cluster] updateStubInMapping -> Allocating a new stub for node 3 with endpoint of osn3.orderer1.moss.com:7063 for channel servic
[orderer.common.cluster] updateStubInMapping -> Deactivating node 3 in channel service1 with endpoint of osn3.orderer1.moss.com:7063 due to TLS
[orderer.common.cluster] func1 -> 2 exists in both old and new membership for channel service1 , skipping its deactivation
[orderer.common.cluster] func1 -> 3 exists in both old and new membership for channel service1 , skipping its deactivation
[orderer.common.cluster] Configure -> Exiting
[orderer.consensus.etcdraft] start -> Starting raft node as part of a new channel channel=service1 node=1
[orderer.consensus.etcdraft] becomeFollower -> 1 became follower at term 0 channel=service1 node=1
[orderer.consensus.etcdraft] newRaft -> newRaft 1 [peers: [], term: 0, commit: 0, applied: 0, lastindex: 0, lastterm: 0] channel=service1 node=1
[orderer.consensus.etcdraft] becomeFollower -> 1 became follower at term 1 channel=service1 node=1
[orderer.consensus.etcdraft] apply -> Applied config change to add node 1, current nodes in channel: [1 2 3] channel=service1 node=1
[orderer.consensus.etcdraft] apply -> Applied config change to add node 2, current nodes in channel: [1 2 3] channel=service1 node=1
[orderer.consensus.etcdraft] apply -> Applied config change to add node 3, current nodes in channel: [1 2 3] channel=service1 node=1
[orderer.consensus.etcdraft] Step -> 1 [logterm: 1, index: 3, vote: 0] cast MsgPreVote for 2 [logterm: 1, index: 3] at term 1 channel=service1 n
[orderer.consensus.etcdraft] Step -> 1 [term: 1] received a MsgVote message with higher term from 2 [term: 2] channel=service1 node=1
[orderer.consensus.etcdraft] becomeFollower -> 1 became follower at term 2 channel=service1 node=1
[orderer.consensus.etcdraft] Step -> 1 [logterm: 1, index: 3, vote: 0] cast MsgVote for 2 [logterm: 1, index: 3] at term 2 channel=service1 node
[orderer.consensus.etcdraft] run -> raft.node: 1 elected leader 2 at term 2 channel=service1 node=1
[orderer.consensus.etcdraft] run -> Raft leader changed: 0 -> 2 channel=service1 node=1
```

네트워크 & 채널 – 5. Peer Join

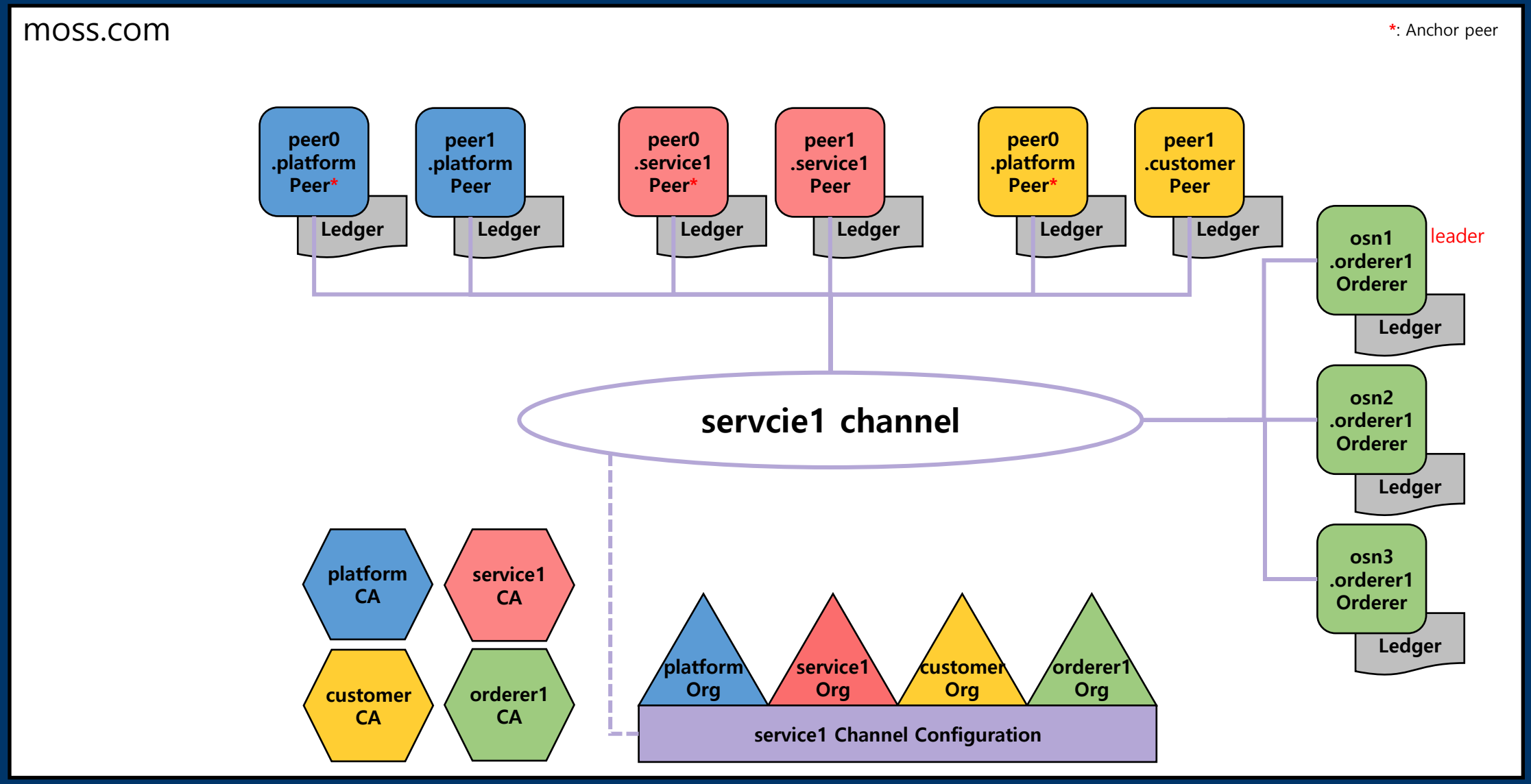


네트워크 & 채널 – 5. Peer Join

platform 조직 peer0 피어 Join

```
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_TLS_ROOTCERT_FILE=./organizations/peerOrganizations/platform.moss.com/tlsca/tlsca.platform.moss.com-cert.pem
export CORE_PEER_LOCALMSPID=platformMSP
export CORE_PEER_MSPCONFIGPATH=./organizations/peerOrganizations/platform.moss.com/users/Admin@platform.moss.com/msp
export CORE_PEER_ADDRESS=localhost:8060

peer channel join -b ./channel-artifacts/service1.block
```

네트워크 & 채널 – 6. Anchor Peer 설정

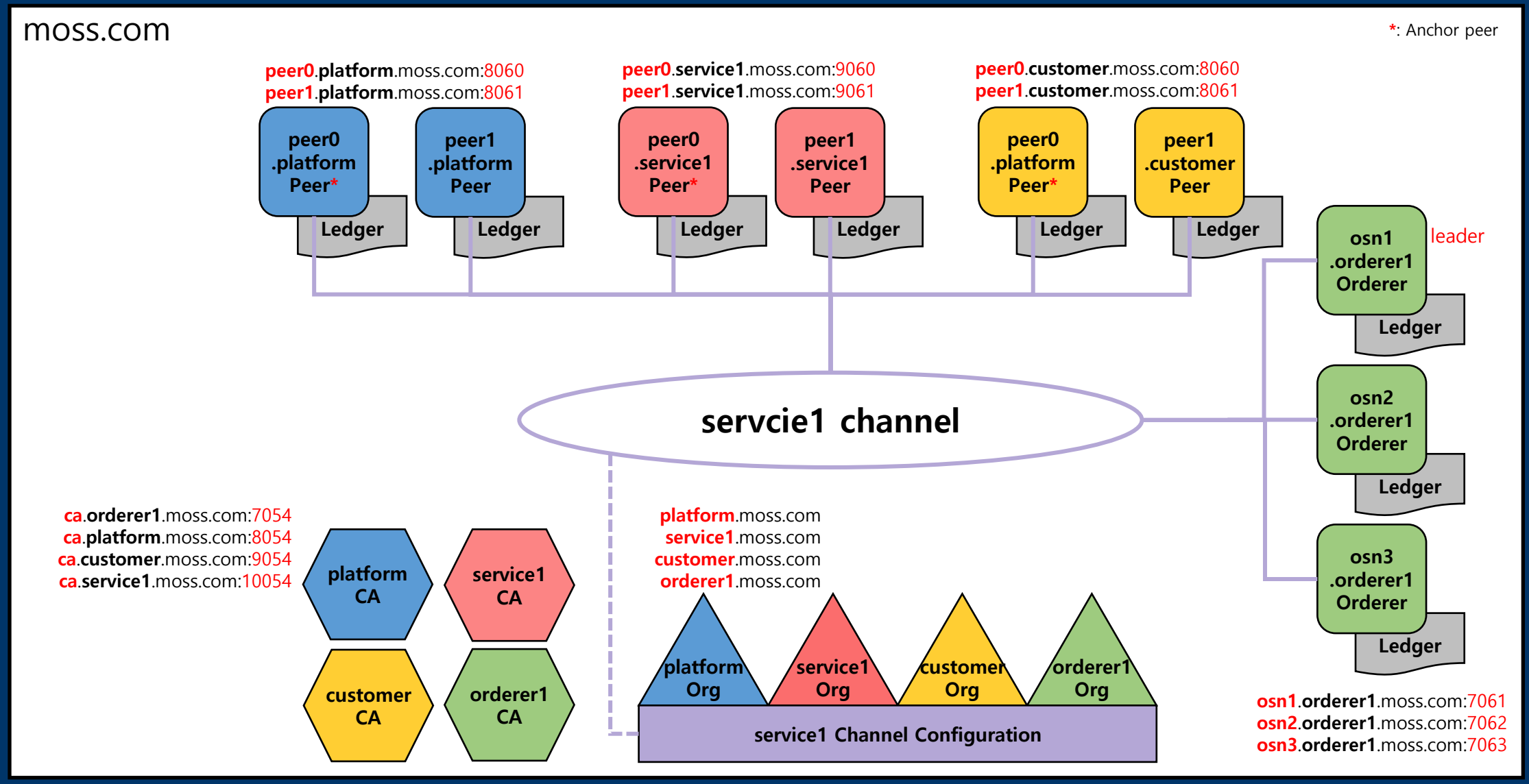


네트워크 & 채널 – 6. Anchor Peer 설정

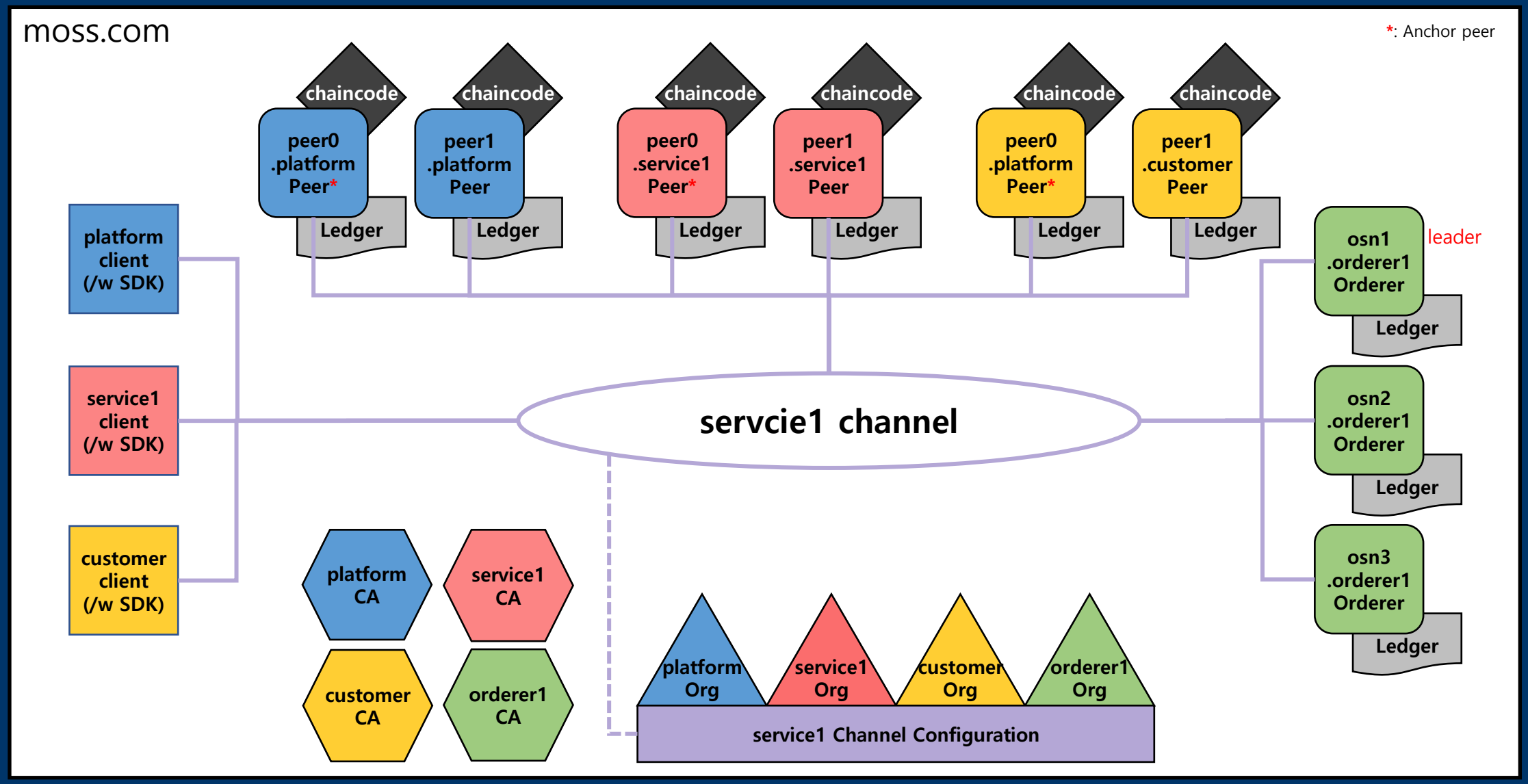
platform 조직 anchor 피어 설정

```
…
jq '.channel_group.groups.Application.groups.platformMSP.values +=
'{
  "AnchorPeers":{
    "mod_policy":"Admins",
    "value":{
      "anchor_peers":[
        {
          "host":"peer0.platform.moss.com",
          "port":8060
        }
      ]
    },
    "version":"0"
  }
}'

config_copy.json > modified_config.json

…

peer channel update -f channel-artifacts/config_update_in_envelope.pb -c ${CHANNEL_NAME} -o ${ORDERER_ADDRESS} --
ordererTLSHostnameOverride ${ORDERER_DOMAIN} --tls --cafile "$ORDERER_CA"
```

네트워크 & 채널 – 도메인 및 port



네트워크 & 채널 – 체인코드와 클라 표시



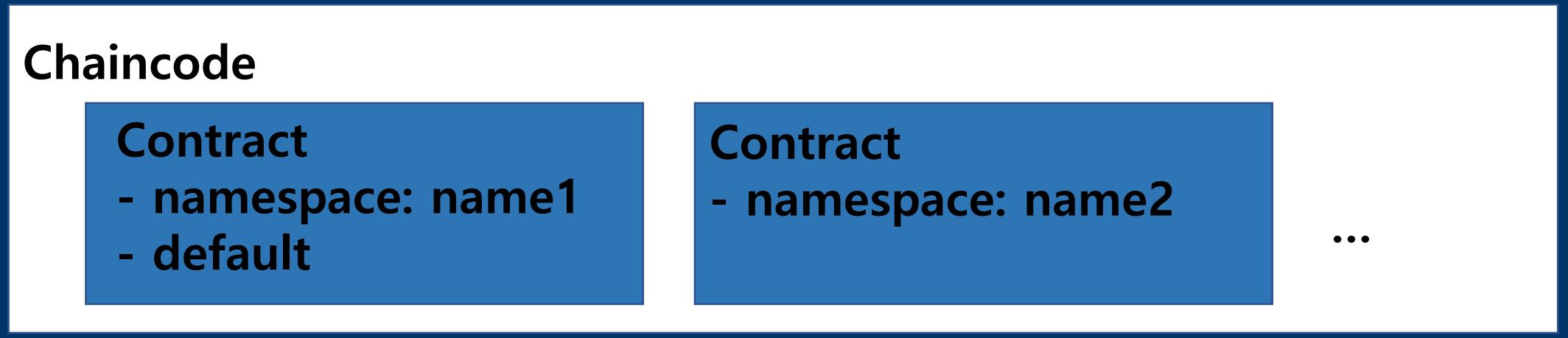
네트워크 & 채널 – 실습

- 로컬 환경에 Fabric 네트워크 구축 – 링크
- 참고
  - [공식문서] Getting Started - Install
  - [공식문서] Getting Started – Run Fabric, Running a Test Network
  - Hyperledger Fabric
  - What's new in v2.x
  - 네트워크 형성 과정
  - test-network
  - test-network 분석

체인코드(Chaincode)

Fabric의 스마트 컨트랙트

- 배포 단위 (Channel, Name, Version, Sequence, Package Id)
  - Channel
  - Name
  - Version: {major}.{minor}, 예) 1.0
  - Sequence: 동일 ChaincodeName commit 마다 1증가, 예) 1
  - Package Id: {name}_{version}:{package file hash}
    simple_1.0:88b86c7a852ce5ce7ee2e9ada9763fdd27a812d3e0dd6cb478baa9c53788f6af
- State 관리 범위 : 동일 Chaincode 내 Contract들은 State 공유



체인코드 - Chaincode

Chaincode는 Chaincode Interface를 구현한 것

- Init : 초기화
- Invoke : 트랜잭션
  - SUBMIT : 추가,수정,삭제 (이전 INVOKE)
  - EVALUATE : 조회 (이전 QUERY)

```
/**
 * Defines methods that all chaincodes must implement.
 */
public interface Chaincode {
    /**
     * Called during an instantiate transaction after the container has been
     * established, allowing the chaincode to initialize its internal data.
     *
     * @param stub the chaincode stub
     * @return the chaincode response
     */
    Response init(ChaincodeStub stub);

    /**
     * Called for every Invoke transaction. The chaincode may change its state
     * variables.
     *
     * @param stub the chaincode stub
     * @return the chaincode response
     */
    Response invoke(ChaincodeStub stub);
}
```

체인코드 - ChaincodeStub

Ledger와 상호작용을 위한 Interface

함 수	설 명
GetCreator()	Client Identity 조회
GetChannelId()	채널 이름 조회
GetMspId()	MSP Id (조직) 조회
GetFunction()	호출 함수
GetArgs()	호출 인자 목록
CreateCompositeKey(objectType, attrs...)	복합키 생성
GetState(key)	상태 조회
GetStateByRange(startKey, endKey)	상태 범위 조회
PutState(key, value)	상태 수정 (미존재 시 추가)
DelState(key)	상태 삭제
SetEvent(name, payload)	이벤트 Emit (다중 호출시 마지막 event로 overwrite)
InvokeChaincode(chaincodeName, args, channel)	다른 체인코드 호출 (다른 채널은 조회만 가능)

체인코드 - ChaincodeStub

```java
/**
 * An object which manages the transaction context, provides access to state variables, and supports calls to
other chaincode implementations.
 */
public interface ChaincodeStub {
    byte[] getCreator();
    String getChannelId();
    String getMspId();
    String getFunction();
    List<byte[]> getArgs();
    CompositeKey createCompositeKey(String objectType, String... attributes);
    byte[] getState(String key);
    QueryResultsIterator<KeyValue> getStateByRange(String startKey, String endKey);
    void putState(String key, byte[] value);
    void delState(String key);
    void setEvent(String name, byte[] payload);
    Response invokeChaincode(String chaincodeName, List<byte[]> args, String channel);
}
```

체인코드 - Key & CompositeKey

- Key: String
- CompositeKey: 복합키
  - Mapping처리를 위해서 사용
  - 포맷: NAMESPACE + objectType + DELIMITER + (attribute + DELIMITER)*
    - MIN_CODE_POINT = 0x000000
    - DELIMITER = MIN_CODE_POINT
    - NAMESPACE = DELIMITER
  - 사용 예

```
CompositeKey key = new CompositeKey("tokens", "1");
stub.putStringState(key.toString(), "tokenValue");
```

체인코드 - Key & CompositeKey

```java
public class CompositeKey {
    private static final String DELIMITER = new String(Character.toChars(Character.MIN_CODE_POINT));
    public static final String NAMESPACE = DELIMITER;

    private final String objectType;
    private final List<String> attributes;
    private final String compositeKey;

    public CompositeKey(final String objectType, final String... attributes) {
        this(objectType, attributes == null ? Collections.emptyList() : Arrays.asList(attributes));
    }

    public CompositeKey(final String objectType, final List<String> attributes) {
        // ...
        this.objectType = objectType;
        this.attributes = attributes;
        this.compositeKey = generateCompositeKeyString(objectType, attributes);
    }

    private String generateCompositeKeyString(final String objectType, final List<String> attributes) {
        // ...
        if (attributes == null || attributes.isEmpty()) {
            return NAMESPACE + objectType + DELIMITER;
        }
        // ...
        // return NAMESPACE + objectType + DELIMITER + (attribute + DELIMITER)*
        return attributes.stream().collect(joining(DELIMITER, NAMESPACE + objectType + DELIMITER, DELIMITER));
    }

    @Override
    public String toString() {
        return compositeKey;
    }
}
```

체인코드 - Response

```
public interface Chaincode {
    /**
     * Wrapper around protobuf Response, contains status, message and payload.
     * Object returned by call to {@link #init(ChaincodeStub)}
     * and{@link #invoke(ChaincodeStub)}
     */
    class Response {

        private final int statusCode;
        private final String message;
        private final byte[] payload;

        /**
         * {@link Response} status enum.
         */
        public enum Status {
            SUCCESS(200),
            ERROR_THRESHOLD(400),
            INTERNAL_SERVER_ERROR(500);
        }
    }
}
```

편의 함수

- ResponseUtils.newSuccessResponse(message, payload)
- ResponseUtils.newErrorResponse(message, payload)

체인코드 - ChaincodeException

컨트랙트 에러 발생시 사용해야하는 Exception
payload가 Response로 전달된다. 이 때 statusCode는 500이 반환된다.

```
/**
 * Contracts should use {@code ChaincodeException} to indicate when an error
 * occurs in Smart Contract logic.
 *
 * <p>
 * When a {@code ChaincodeException} is thrown an error response will be
 * returned from the chaincode container containing the exception message and
 * payload, if specified.
 *
 * <p>
 * {@code ChaincodeException} may be extended to provide application specific
 * error information. Subclasses should ensure that {@link #getPayload} returns
 * a serialized representation of the error in a suitable format for client
 * applications to process.
 */
public class ChaincodeException extends RuntimeException {
    private byte[] payload;
    ...
}
```

```
// ResponseUtils.newErrorResponse(Throwable throwable)
if (throwable instanceof ChaincodeException) {
    message = throwable.getMessage();
    payload = ((ChaincodeException) throwable).getPayload();
    return new Chaincode.Response(INTERNAL_SERVER_ERROR, message, payload);
```

체인코드 - SimpleChaincode

```java
public class SimpleChaincode extends ChaincodeBase {
    @Override
    public Response init(ChaincodeStub stub) {
        stub.putStringState("key", "initValue");
        return ResponseUtils.newSuccessResponse();
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
        String function = stub.getFunction();
        if ("get".equals(function)) {
            return get(stub);
        } else if ("set".equals(function)) {
            return set(stub, stub.getStringArgs().get(0));
        } else {
            throw new ChaincodeException("Undefined contract method called.");
        }
    }

    private Response get(ChaincodeStub stub) {
        String value = stub.getStringState("key");
        return ResponseUtils.newSuccessResponse(value.getBytes(StandardCharsets.UTF_8));
    }

    private Response set(ChaincodeStub stub, String value) {
        stub.putStringState("key", value);
        return ResponseUtils.newSuccessResponse();
    }
}
```

체인코드 - ContractRouter

- ChaincodeBase를 상속받아 Annotation 기반으로 동작
- Init : 미사용 - Init 여부는 approve 시 --init-required 옵션으로 지정

```
/**
 * Router class routes Init/Invoke requests to contracts. Implements
 * {@link org.hyperledger.fabric.shim.Chaincode} interface.
 * @see ContractInterface
 */
public final class ContractRouter extends ChaincodeBase {

    @Override
    public Response invoke(final ChaincodeStub stub) {
        return processRequest(stub);
    }

    @Override
    public Response init(final ChaincodeStub stub) {
        return processRequest(stub);
    }

    /**
     * Main method to start the contract based chaincode.
     */
    public static void main(final String[] args) throws Exception {
        final org.hyperledger.fabric.contract.ContractRouter cfc =
            new org.hyperledger.fabric.contract.ContractRouter(args);
        cfc.findAllContracts();
        ...
    }
}
```

체인코드 - ContractRouter's Contract

- 필수
  - ContractInterface : Contract 탐색 및 트랜잭션 hook 제공
  - @Contract : Contract 지정
- 옵션
  - @Default : 기본 Contract 지정
  - @Transaction : 트랜잭션 함수 지정
    - method 이름이 functionName
    - 첫번째 인자에 Context 제공
    - 두번째 인자부터 Args Binding 제공
    - 반환 값 Json으로 Serialize 후, Response 자동 생성

체인코드 - ContractInterface

```java
public interface ContractInterface {
    /**
     * Create context from {@link ChaincodeStub}.
     * Default impl provided, but can be
     * overwritten by contract
     */
    default Context createContext(final ChaincodeStub stub) {
        return ContextFactory.getInstance().createContext(stub);
    }

    /**
     * Invoked for any transaction that does not exist.
     * This will throw an exception. If you wish to alter the exception thrown or if
     * you wish to consider requests for transactions that don't exist as not an
     * error, subclass this method.
     */
    default void unknownTransaction(final Context ctx) {
        throw new ChaincodeException("Undefined contract method called");
    }

    /**
     * Invoked once before each transaction.
     * Any exceptions thrown will fail the transaction, and neither the required
     * transaction or the {@link #afterTransaction(Context, Object)} will be called
     */
    default void beforeTransaction(final Context ctx) { }

    /**
     * Invoked once after each transaction.
     * Any exceptions thrown will fail the transaction.
     */
    default void afterTransaction(final Context ctx, final Object result) { }
}
```

체인코드 - Context

- ChaincodeStub 제공
- ClientIdentity 제공 - ChaincodeStub.getCreator() 값 이용

```
public class Context {
    /**
     *
     * @return ChaincodeStub instance to use
     */
    public ChaincodeStub getStub() {
        return this.stub;
    }

    /**
     *
     * @return ClientIdentity object to use
     */
    public ClientIdentity getClientIdentity() {
        return this.clientIdentity;
    }
}
```

체인코드 - SimpleContract

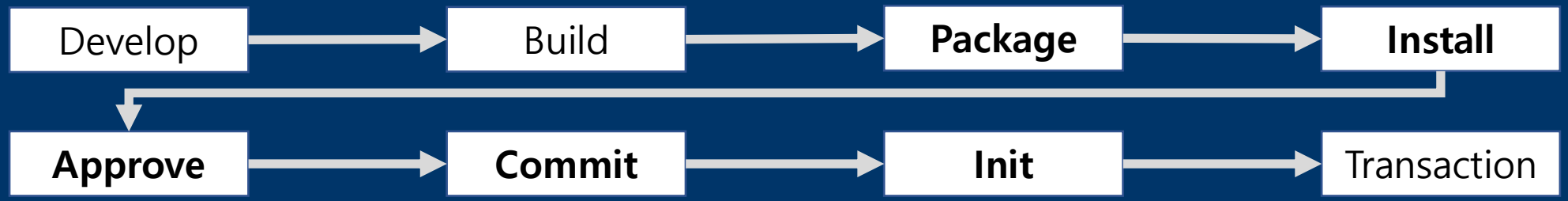
```java
@Default
@Contract(name = "simple")
public class SimpleContract implements ContractInterface {

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public void init(Context context) {
        ChaincodeStub stub = context.getStub();
        String isInit = stub.getStringState("isInit");
        if ("true".equals(isInit)) {
            throw new ChaincodeException("already initialized.");
        }
        stub.putStringState("key", "initValue");
        stub.putStringState("isInit", "true");
    }

    @Transaction(intent = Transaction.TYPE.EVALUATE)
    public String get(Context context) {
        return context.getStub().getStringState("key");
    }

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public void set(Context context, String value) {
        context.getStub().putStringState("key", value);
    }
}
```

Source Code

체인코드 - SimpleContract

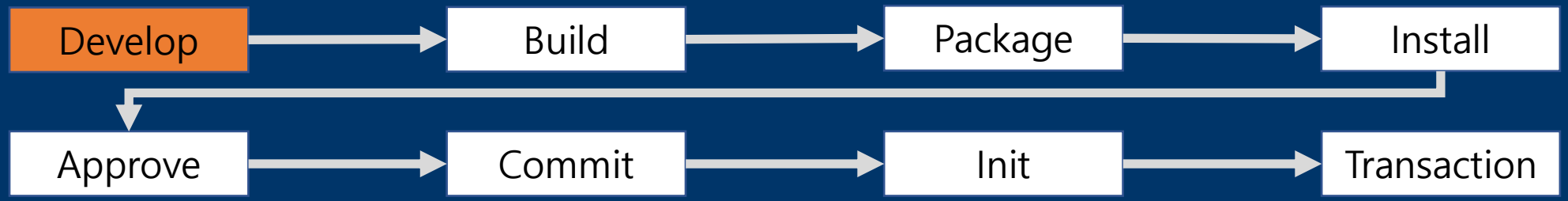
```java
public class SimpleContract extends ChaincodeBase {
    @Override
    public Response init(ChaincodeStub stub) {
        stub.putStringState("key", "initValue");
        return ResponseUtils.newSuccessResponse();
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
        String function = stub.getFunction();
        if ("get".equals(function)) {
            return get(stub);
        } else if ("set".equals(function)) {
            return set(stub, stub.getStringArgs().get(0));
        } else {
            throw new ChaincodeException("wrong function.");
        }
    }

    private Response get(ChaincodeStub stub) {
        String value = stub.getStringState("key");
        return ResponseUtils.newSuccessResponse(value.getBytes(StandardCharsets.UTF_8));
    }

    private Response set(ChaincodeStub stub, String value) {
        stub.putStringState("key", value);
        return ResponseUtils.newSuccessResponse();
    }
}
```

체인코드 - 배포(chaincode lifecycle)



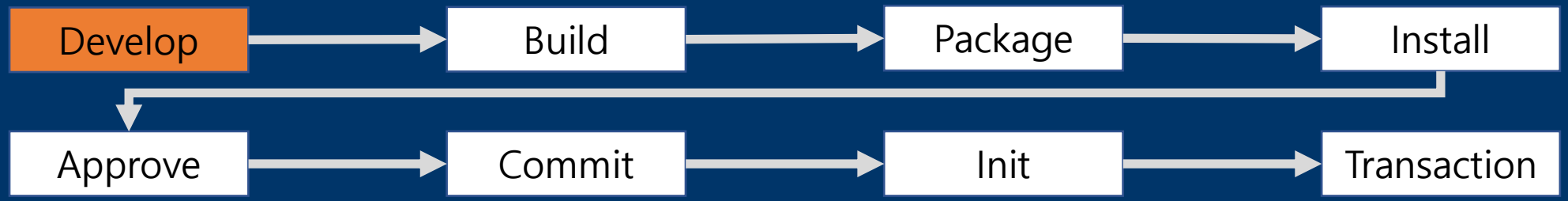
- Develop : 체인코드 개발
- Build : 언어별 빌드
- Package : tar.gz 파일로 Package
- Install : 체인코드를 운영할 peer 들에 설치
- Approve : 각 조직이 chaincode에 대해 Approve (기본 정책: 과반수)
- Commit : 체인코드를 peer에서 container로 실행
- Init (옵션) : Approve 시 --require-init 옵션을 사용한 경우 invoke --isInit 옵션으로 초기화
- Transaction : 트랜잭션 제출

체인코드 - 배포(chaincode lifecycle)



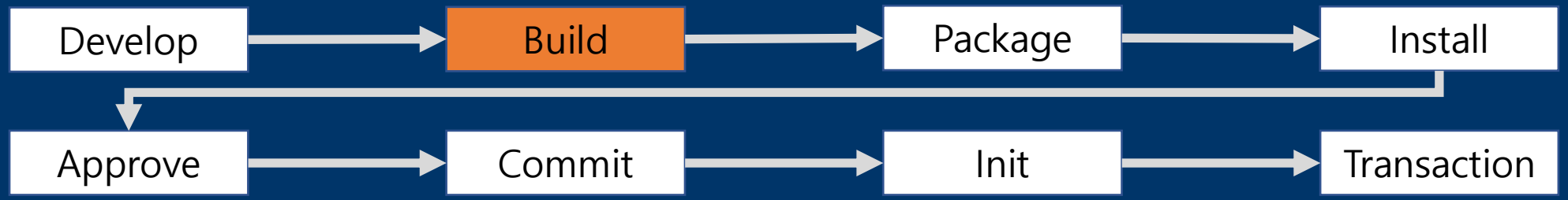

```java
// Java
@Default
@Contract(name = "simple")
public class SimpleContract implements ContractInterface {

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public void init(Context context) {
        ChaincodeStub stub = context.getStub();
        stub.putStringState("key", "initValue");
    }

    @Transaction(intent = Transaction.TYPE.EVALUATE)
    public String get(Context context) {
        return context.getStub().getStringState("key");
    }

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public void set(Context context, String value) {
        context.getStub().putStringState("key", value);
    }
}
```

체인코드 - 배포(chaincode lifecycle)




```typescript
// TypeScript
import {Context, Contract, Transaction} from 'fabric-contract-api'

export class Counter extends Contract {

    @Transaction(true)
    public async init(ctx: Context): Promise<void> {
        await ctx.stub.putState('key', Buffer.from('initValue'));
    }

    @Transaction()
    public async get(ctx: Context): Promise<string> {
        const bytes = await ctx.stub.getState('key');
        return bytes.toString();
    }

    @Transaction(true)
    public async set(ctx: Context, value: string): Promise<void> {
        await ctx.stub.putState('key', Buffer.from(value));
    }
}
```

체인코드 - 배포(chaincode lifecycle)

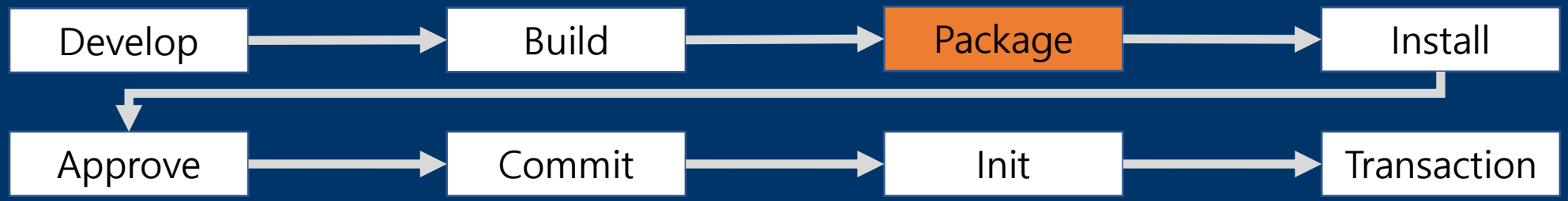


- Java : ./gradlew installDist
- javascript : npm install
- typescript : npm install && tsc
- go : GO111MODULE=on go mod vendor

작업의 편의를 위하여 체인코드 소스코드 경로를 환경변수로 지정

```
export CC_SRC_PATH=${PWD}
```

체인코드 - 배포(chaincode lifecycle)



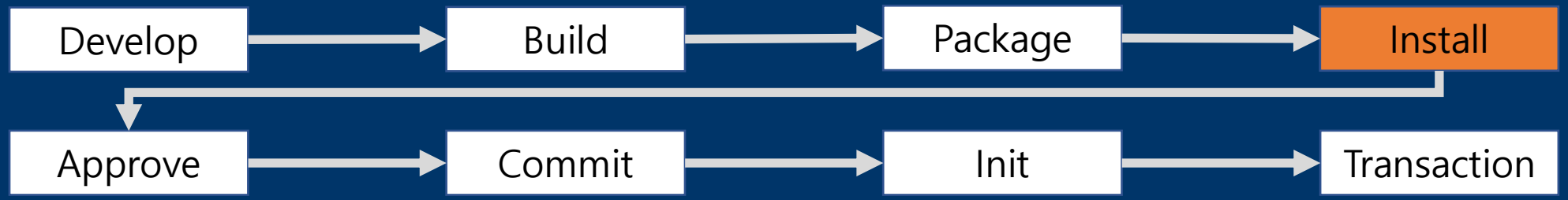
체인코드 배포에 필요한 환경변수 설정

```
cd "$FABRIC_NETWORK_HOME"
export CC_NAME=simple
export CC_SRC_PATH=${CC_SRC_PATH}
export CC_RUNTIME_LANGUAGE=java # java, node, golang
export CC_VERSION=1.0
export CC_SEQUENCE=1  # commit 마다 1증가
export CC_INIT_FCN=init
export CC_END_POLICY=""
export CC_COLL_CONFIG=""
export INIT_REQUIRED="--init-required"
```

Package : simple.tar.gz 파일 생성

```
peer lifecycle chaincode package
  ${CC_NAME}.tar.gz # simple.tar.gz
  --path "${CC_SRC_PATH}/build/install/${CC_NAME}" # ${CC_SRC_PATH}/build/install/simple
  --lang ${CC_RUNTIME_LANGUAGE} # java
  --label ${CC_NAME}_${CC_VERSION} # simple_1.0
```

체인코드 - 배포(chaincode lifecycle)



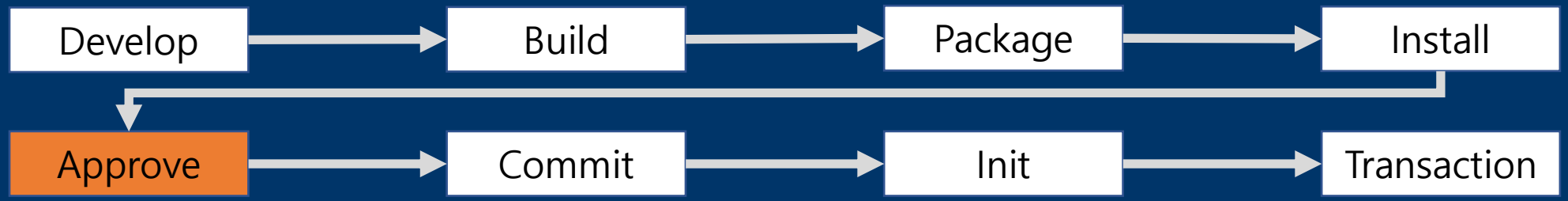
체인코드를 운영할 peer들에 chaincode install (여기서는 조직의 peer0에만 install)

```
. ./scripts/setPlatformPeer0.sh  # platform 조직, peer0 피어와 통신하도록 설정
Peer lifecycle chaincode install ${CC_NAME}.tar.gz  # simple.tar.gz

. ./scripts/setCustomerPeer0.sh  # customer 조직, peer0 피어와 통신하도록 설정
Peer lifecycle chaincode install ${CC_NAME}.tar.gz

. ./scripts/setService1Peer0.sh  # service1 조직, peer0 피어와 통신하도록 설정
peer lifecycle chaincode install ${CC_NAME}.tar.gz
```

PackageId 계산 및 환경변수로 지정

```
peer lifecycle chaincode calculatepackageid ${CC_NAME}.tar.gz >&log.txt  # simple.tar.gz
PACKAGE_ID=$(sed -n "1p" log.txt)
echo $PACKAGE_ID  # simple_1.0:88b86c7a852ce5ce7ee2e9ada9763fdd27a812d3e0dd6cb478baa9c53788f6af
```

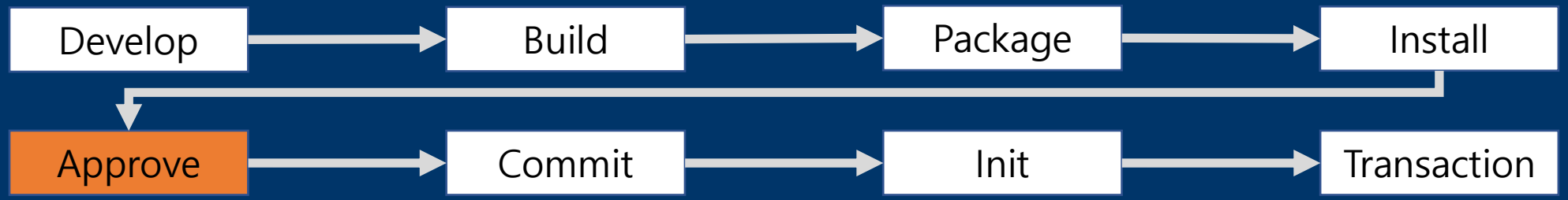
체인코드 - 배포(chaincode lifecycle)



조직별로 설치된 chaincode를 Approve 한다

```
. ./scripts/setPlatformPeer0.sh  # platform 조직, peer0 피어와 통신하도록 셋팅
peer lifecycle chaincode approveformyorg -o ${ORDERER_ADDRESS} --ordererTLSHostnameOverride ${ORDERER_DOMAIN}
--tls --cafile "$ORDERER_CA"
--channelID $CHANNEL_NAME --name ${CC_NAME} --version ${CC_VERSION}
--package-id ${PACKAGE_ID} --sequence ${CC_SEQUENCE} ${INIT_REQUIRED} ${CC_END_POLICY} ${CC_COLL_CONFIG}

# peer lifecycle chaincode approveformyorg -o localhost:7061 --ordererTLSHostnameOverride osn1.orderer1.moss.com
--tls --cafile ./…/osn1.orderer1.moss.com/…/tlsca.orderer1.moss.com-cert.pem
--channelID service1 --name counter-java --version 1.0
--package-id simple_1.0:88b86c7a852ce5ce7ee2e9ada9763fdd27a812d3e0dd6cb478baa9c53788f6af --sequence 1 --init-required

. ./scripts/setCustomerPeer0.sh  # customer 조직, peer0 피어와 통신하도록 셋팅
peer lifecycle chaincode approveformyorg …

. ./scripts/setService1Peer0.sh  # service1 조직, peer0 피어와 통신하도록 셋팅
peer lifecycle chaincode approveformyorg …
```

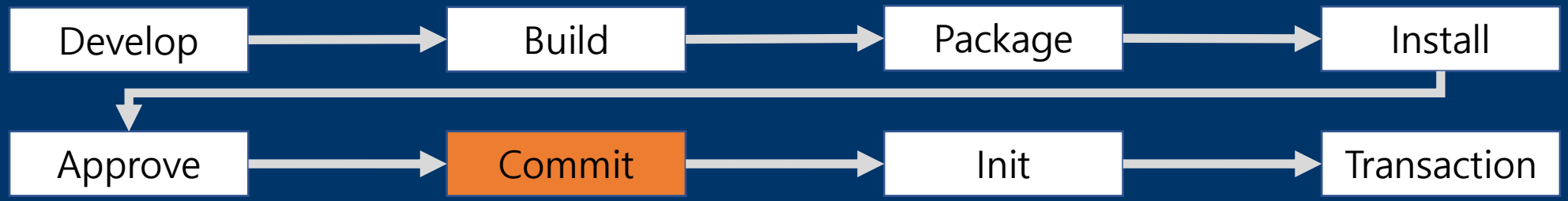
체인코드 - 배포(chaincode lifecycle)



Approve 확인

```
peer lifecycle chaincode checkcommitreadiness
--channelID $CHANNEL_NAME --name ${CC_NAME} --version ${CC_VERSION} --sequence ${CC_SEQUENCE}
${INIT_REQUIRED} ${CC_END_POLICY} ${CC_COLL_CONFIG} --output json >&log.txt

# peer lifecycle chaincode checkcommitreadiness
--channelID service1 --name simple --version 1.0 --sequence 1
--init-required --output json

cat log.txt

{
        "approvals": {
                "customerMSP": true,
                "service1MSP": true,
                "platformMSP": true
        }
}
```

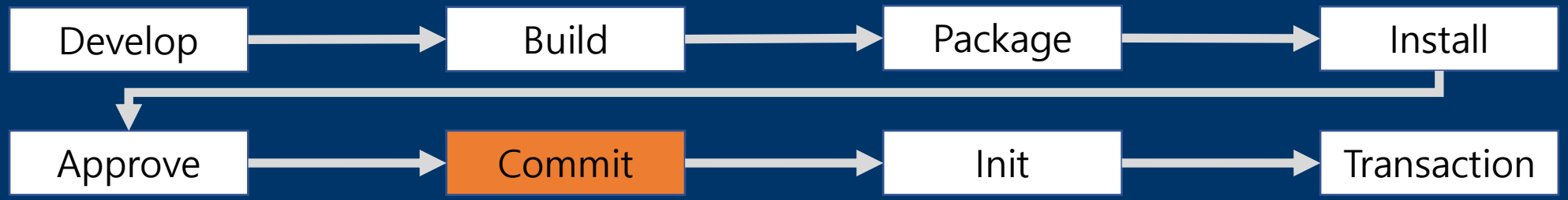
체인코드 - 배포(chaincode lifecycle)



Chaincode를 설치한 peer에 commit

```
PEER_CONN_PARAMS=(--peerAddresses localhost:8060 --tlsRootCertFiles "./…/platform.moss.com/…/tlsca.platform.moss.com-cert.pem")
PEER_CONN_PARAMS+=(--peerAddresses localhost:9060 --tlsRootCertFiles "./…/customer.moss.com/…/tlsca.customer.moss.com-cert.pem")
PEER_CONN_PARAMS+=(--peerAddresses localhost:10060 --tlsRootCertFiles "./…/service1.moss.com/…/tlsca.service1.moss.com-cert.pem" )

peer lifecycle chaincode commit -o ${ORDERER_ADDRESS} --ordererTLSHostnameOverride ${ORDERER_DOMAIN}
--tls --cafile "$ORDERER_CA"
--channelID $CHANNEL_NAME --name ${CC_NAME}
"${PEER_CONN_PARAMS[@]}"
--version ${CC_VERSION} --sequence ${CC_SEQUENCE} ${INIT_REQUIRED} ${CC_END_POLICY} ${CC_COLL_CONFIG}

# peer lifecycle chaincode commit -o localhost:7061 --ordererTLSHostnameOverride osn1.orderer1.moss.com
--tls --cafile ./…/osn1.orderer1.moss.com/…/tlsca.orderer1.moss.com-cert.pem
--channelID service1 --name simple
--peerAddresses localhost:8060 --tlsRootCertFiles ./…/tlsca.platform.moss.com-cert.pem
--peerAddresses localhost:9060 --tlsRootCertFiles ./…/tlsca.customer.moss.com-cert.pem
--peerAddresses localhost:10060 --tlsRootCertFiles ./…/tlsca.service1.moss.com-cert.pem
--version 1.0 --sequence 4 --init-required
```

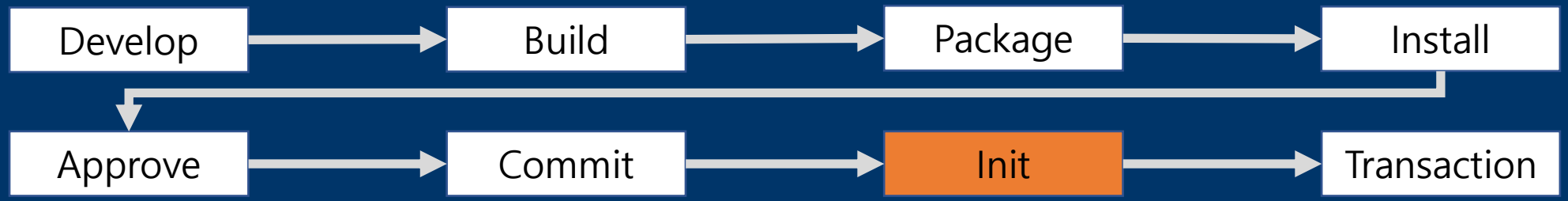
체인코드 - 배포(chaincode lifecycle)



Commit 성공 후 chaincode container 실행 확인

```
$ docker container ls --format "table {{.ID}}\t{{.Names}}"
CONTAINER ID   NAMES
ed28be4910e9   dev-peer0.platform.moss.com-simple_1.0-8e5fd44f9da499d111f79aa122ed76ec30258b048a1ac6fa30233b2acdcf8621
2db0ce9b8ea6   dev-peer0.customer.moss.com-simple_1.0-8e5fd44f9da499d111f79aa122ed76ec30258b048a1ac6fa30233b2acdcf8621
0191d729eec8   dev-peer0.service1.moss.com-simple_1.0-8e5fd44f9da499d111f79aa122ed76ec30258b048a1ac6fa30233b2acdcf8621
```

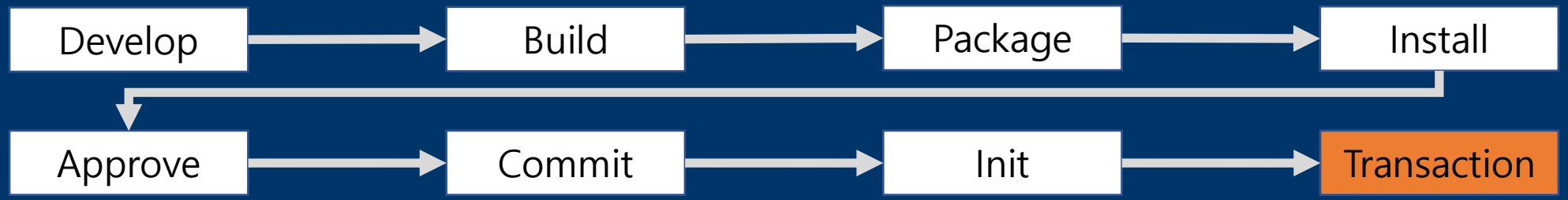
체인코드 - 배포(chaincode lifecycle)



Init 함수 호출 (Approve 시 --init-required 옵션 추가한 경우만)

```
fcn_call='{"function":"'${CC_INIT_FCN}'","Args":[]}' # {"function":"init","Args":[]}

. ./scripts/setPlatformPeer0.sh
peer chaincode invoke -o ${ORDERER_ADDRESS} --ordererTLSHostnameOverride ${ORDERER_DOMAIN}
--tls --cafile "$ORDERER_CA"
-C $CHANNEL_NAME -n ${CC_NAME}
"${PEER_CONN_PARAMS[@]}" --isInit -c ${fcn_call}

# peer chaincode invoke -o localhost:7061 --ordererTLSHostnameOverride osn1.orderer1.moss.com
--tls --cafile ./…/tlsca.orderer1.moss.com-cert.pem
-C service1 -n simple
--peerAddresses localhost:8060 --tlsRootCertFiles ./…/tlsca.platform.moss.com-cert.pem
--peerAddresses localhost:9060 --tlsRootCertFiles ./…/tlsca.customer.moss.com-cert.pem
--peerAddresses localhost:10060 --tlsRootCertFiles ./…/tlsca.service1.moss.com-cert.pem
--isInit -c {"function":"init","Args":[]}
```

체인코드 - 배포(chaincode lifecycle)



체인코드 배포 완료 후 Submit(invoke), Evaluate(query)

```
peer chaincode query -C ${CHANNEL_NAME} -n ${CC_NAME} -c '{"Args":["get"]}'
# initValue

peer chaincode invoke -o ${ORDERER_ADDRESS} --ordererTLSHostnameOverride ${ORDERER_DOMAIN}
--tls --cafile "$ORDERER_CA" -C $CHANNEL_NAME -n ${CC_NAME} "${PEER_CONN_PARAMS[@]}"
-c '{"function":"set","Args":["newVaule"]}'
# 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200

peer chaincode query -C ${CHANNEL_NAME} -n ${CC_NAME} -c '{"Args":["get"]}'
# newValue
```

체인코드 - message

메시지: {"Function":"({namespace}:){function}","Args":[("{arg}")*]}

- Function: 호출할 Contract 및 함수, 포맷: ({namespace}:){function}
- Args: 문자열 인자 목록, 포맷: [("{arg}")*]
- 예) {"Function":"set","Args":["newValue"]

기타 규칙

- Function, Args 키 값은 insensitive(대소문자 구분하지 않음)
- namespace가 지정되지 않으면 @Default Contract가 호출된다.
- Function 값이 없으면 Args의 첫번째 인자가 Function이 된다
- Function이 존재하지 않으면 ContractInterface.unknownTransaction(Context) 호출

체인코드 - message

SimpleContract

```
@Default
@Contract(name = "simple")
public class SimpleContract implements ContractInterface {
    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public void set(Context context, String value) {
        context.getStub().putStringState("key", value);
    }
}
```

아래는 모두 같은 호출 메시지

```
{"Function":"simple:set", "Args":["newValue"]}
{"Function":"set", "Args":["newValue"]}
{"Args":["simple:set", "newValue"]}
{"Args":["set", "newValue"]}
```

체인코드 - fabric-contracts

Solidity OpenZeppelin Contracts와 같은 Contract 개발 라이브러리

crypto.Hash (web3j Hash base)

```
public static byte[] sha256(byte[] input)
public static String sha256String(String utf8String)
```

crypto.Keys (web3j Keys base)

```
public static final String ZeroAddress = "0x0000000000000000000000000000000000000000";
public static String getAddress(PublicKey publicKey) // Ethereum 방식에서 Hash만 Sha256으로 변경
public static String toChecksumAddress(String address) // EIP-55 Checksum 적용
public static boolean validateAddress(String address) // EIP-55 유효성 체크
```

체인코드 - fabric-contracts

util.Json - Json determinism을 위해 Genson 사용

```
public static String serialize(final Object obj)
public static byte[] serializeToBytes(final Object obj)
public static <T> T deserialize(final String json, Class<T> clz)
public static <T> T deserializeFromBytes(final byte[] bytes, Class<T> clz)
```

util.Assert – 비유효시 ChaincodeException 발생

```
public static void require(boolean required, String message)
public static void address(String address)
public static void notZeroAddress(String address)
```

util.SafeMath - Overflow, Underflow시 ChaincodeException 발생

```
public static Long add(Long a, Long b)
public static Long sub(Long a, Long b)
public static Long mul(Long a, Long b)
public static Long div(Long a, Long b)
public static Long mod(Long a, Long b)
```

체인코드 - fabric-contracts

util.State – World State 조작

```java
// 트랜잭션 호출자의 Address 조회
public static String msgSender(final Context context) {
    return Keys.getAddress(context.getClientIdentity().getX509Certificate().getPublicKey());
}

// Long 조회, 값이 존재하지 않으면 0을 반환합니다.
public static Long getLong(final ChaincodeStub stub, final CompositeKey key)
public static Long getLong(final ChaincodeStub stub, final String key) {
    String result = stub.getStringState(key);
    if (result.length() == 0) {
        return 0L;
    }
    return Long.parseLong(result);
}

// 조회
public static <T> T get(final ChaincodeStub stub, final Object key, Class<T> clz)

// 수정, value가 null인 경우 delState를 호출하여 삭제합니다.
public static void put(final ChaincodeStub stub, final Object key, final Object value)


// 이벤트 emit
public static void emit(final ChaincodeStub stub, final Object event) {
    emit(stub, event.getClass().getSimpleName(), event);
}

public static void emit(final ChaincodeStub stub, final String name, final Object event) {
    stub.setEvent(name, Json.serializeToBytes(event));
}
```

체인코드 - fabric-contracts

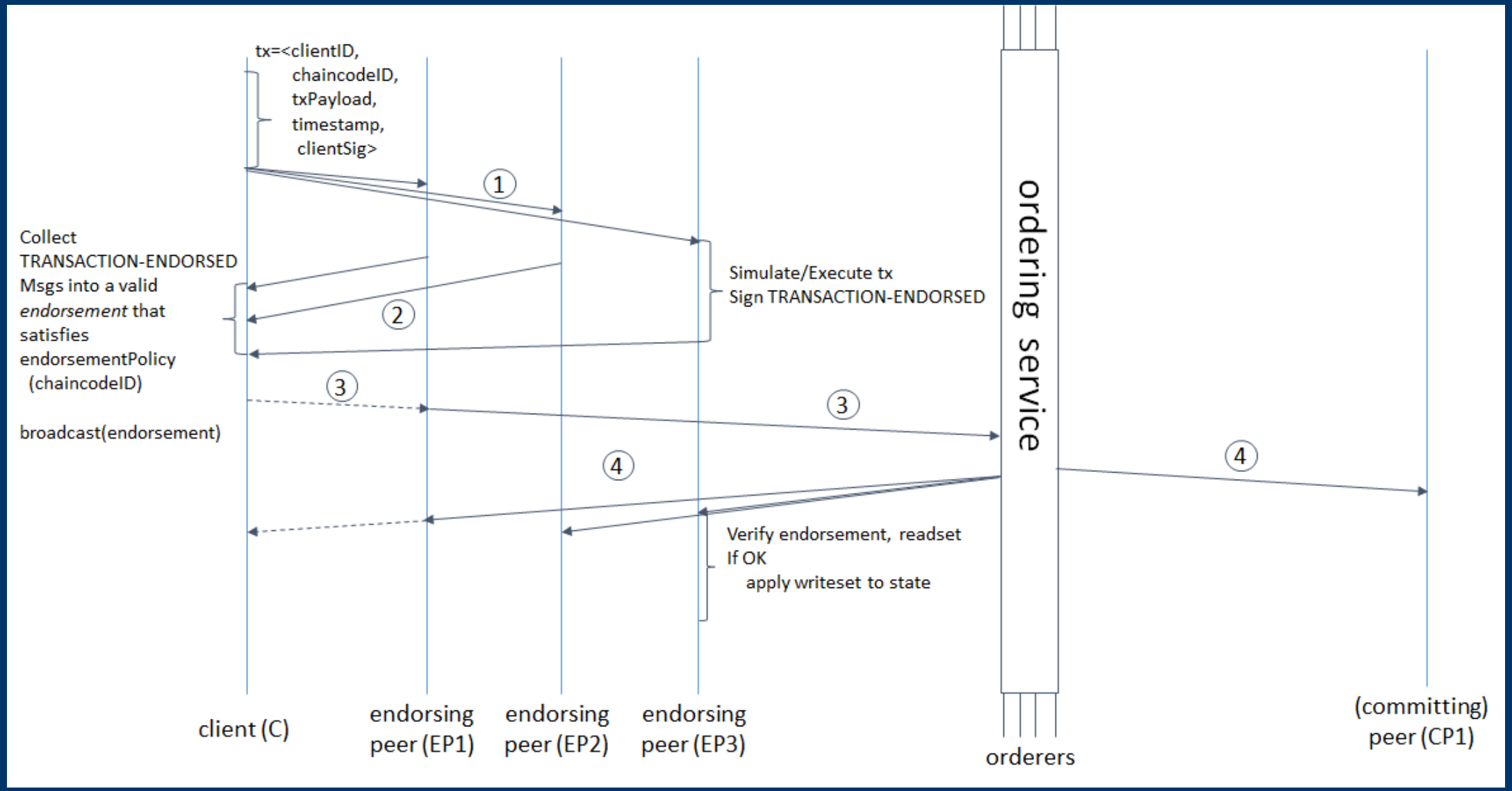
token.erc20.ERC20

```
@Default
@Contract(name = "test-token")
public class TestToken extends ERC20 {

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public Boolean mint(final Context context, final String account, final Long value) {
        Assert.address(account);
        return super._mint(context, account, value);
    }

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public Boolean burn(final Context context, final Long value) {
        String msgSender = State.msgSender(context);
        return super._burn(context, msgSender, value);
    }

    @Transaction(intent = Transaction.TYPE.SUBMIT)
    public Boolean burnFrom(final Context context, final String account, final Long value) {
        Assert.address(account);
        return super._burnFrom(context, account, value);
    }
}
```

체인코드 – 실습

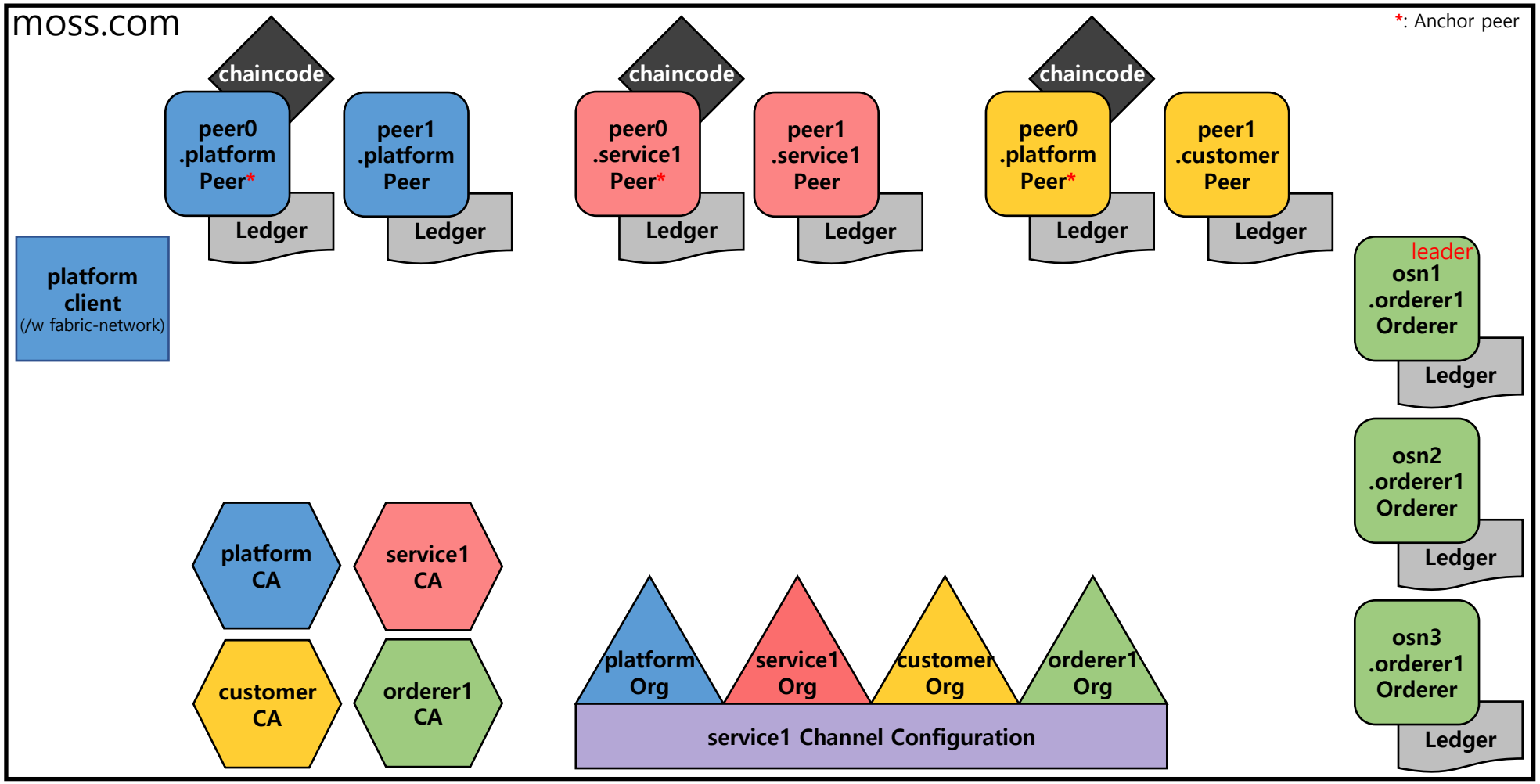
- Hyperledger Fabric Chaincode 개발 – 링크
- 참고
  - [공식문서] Fabric chaincode lifecycle
  - [공식샘플] asset-transfer-basic , asset-transfer-events
  - [공식샘플] token-erc-20, token-erc-721, token-erc-1155
  - fabric-samples, asset-transfer-basic, chaincode 테스트
  - ERC20 샘플 분석

트랜잭션 Flow


tx=<clientID,
chaincodeID,
txPayload,
timestamp,
clientSig>
1
Collect
TRANSACTION-ENDORSED
Msgs into a valid
endorsement that
satisfies
endorsementPolicy
(chaincodeID)
2
Simulate/Execute tx
Sign TRANSACTION-ENDORSED
3
broadcast(endorsement)
3
4
4
Verify endorsement, readset
If OK
apply writeset to state
ordering service
client (C)
endorsing
peer (EP1)
endorsing
peer (EP2)
endorsing
peer (EP3)
orderers
(committing)
peer (CP1)

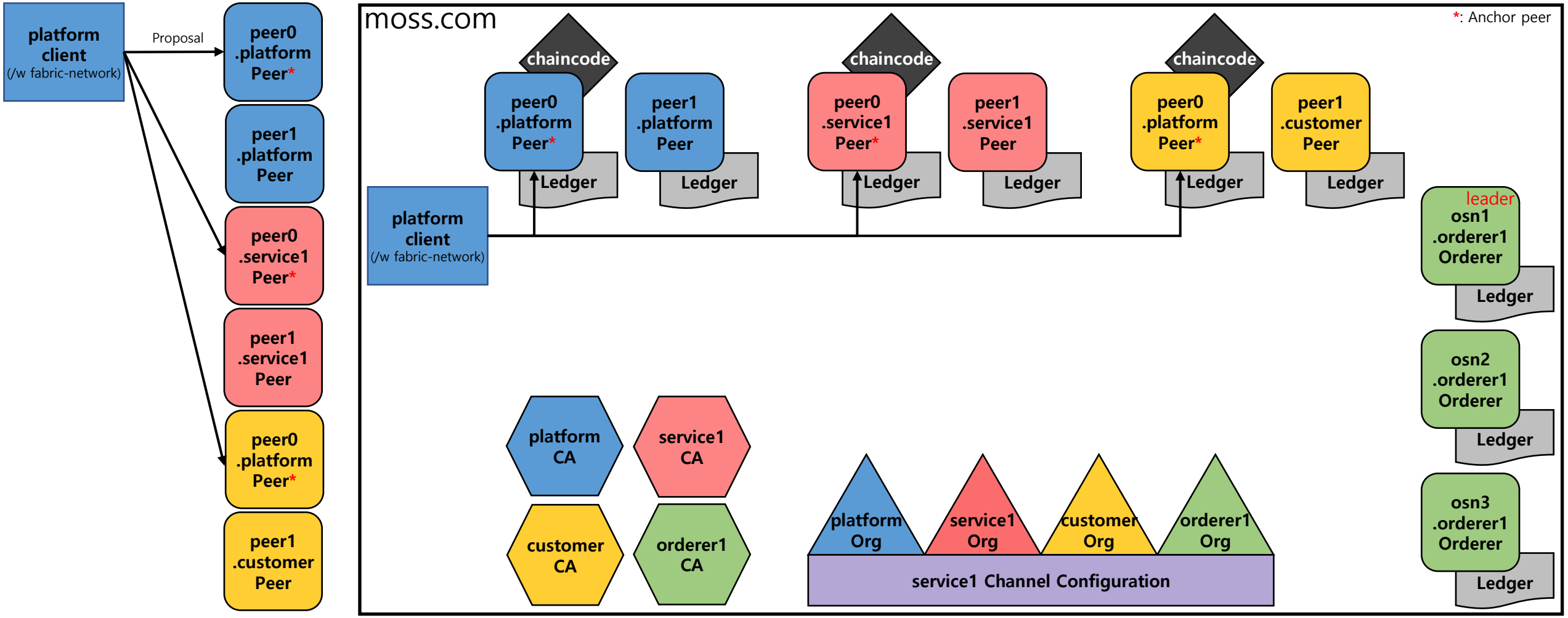
트랜잭션 Flow

0. 각 조직의 peer0에만 chaincode 배포된 상황



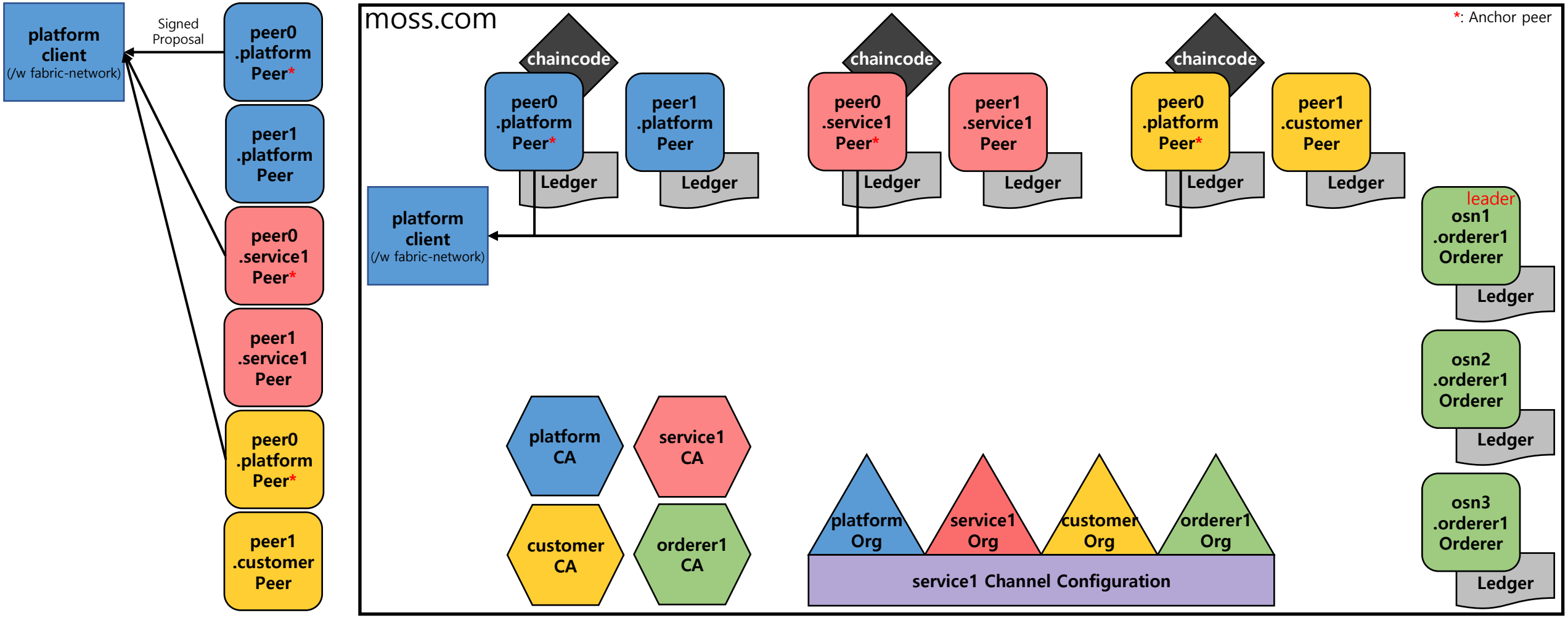
트랜잭션 Flow

1. 트랜잭션 Proposal to Peers



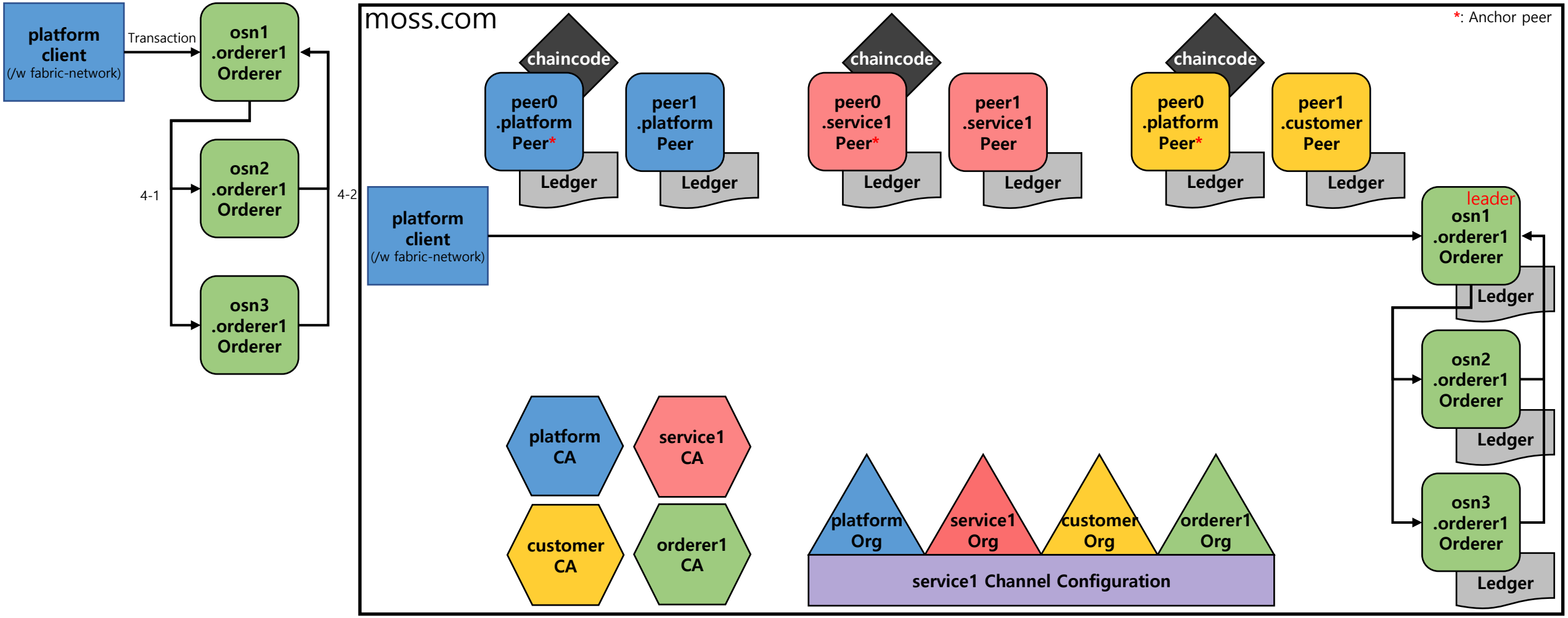
트랜잭션 Flow

2. 트랜잭션 Endorsement by Peers (트랜잭션 실행, 검증, 보증 및 사인)



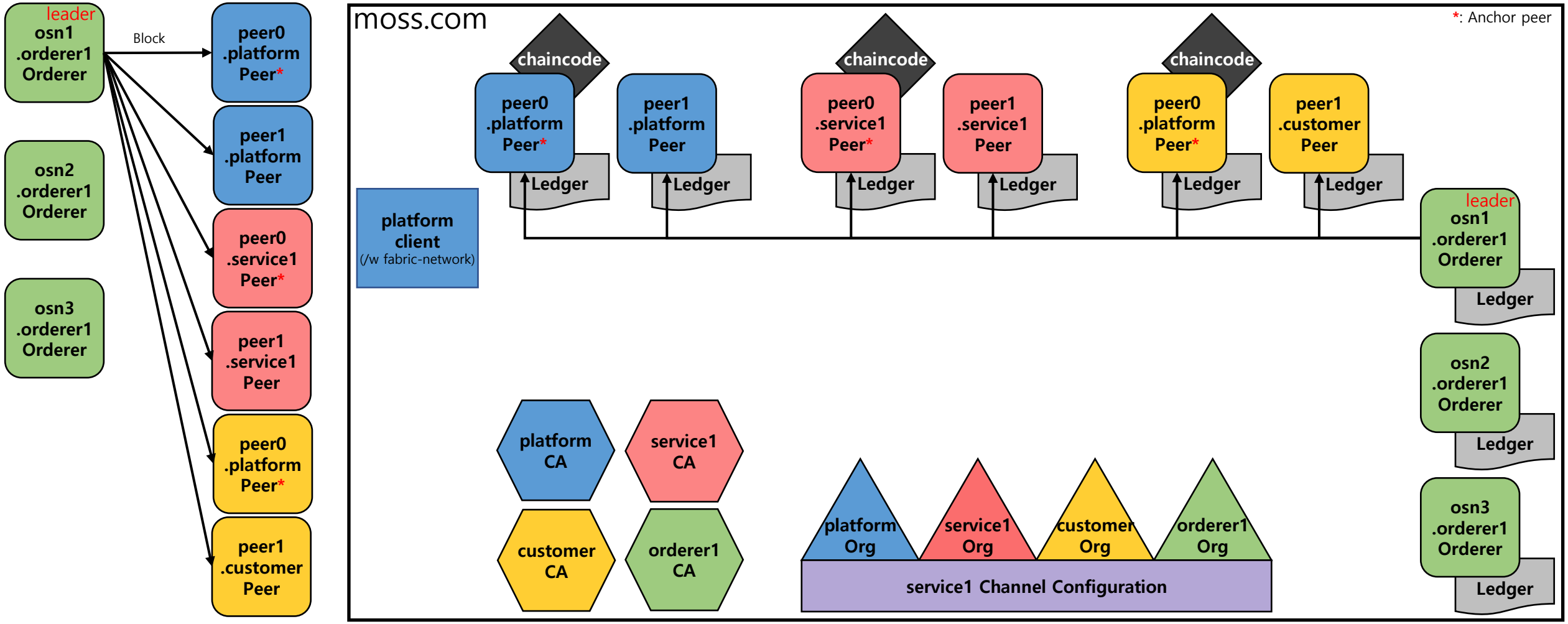
트랜잭션 Flow

3. 트랜잭션을 Ordering 서비스에 제출(submit) (orderer가 leader가 아니면 leader에게 forward)
4. Ordering 서비스는 합의 알고리즘에 따라 트랜잭션 정렬 및 블록 생성



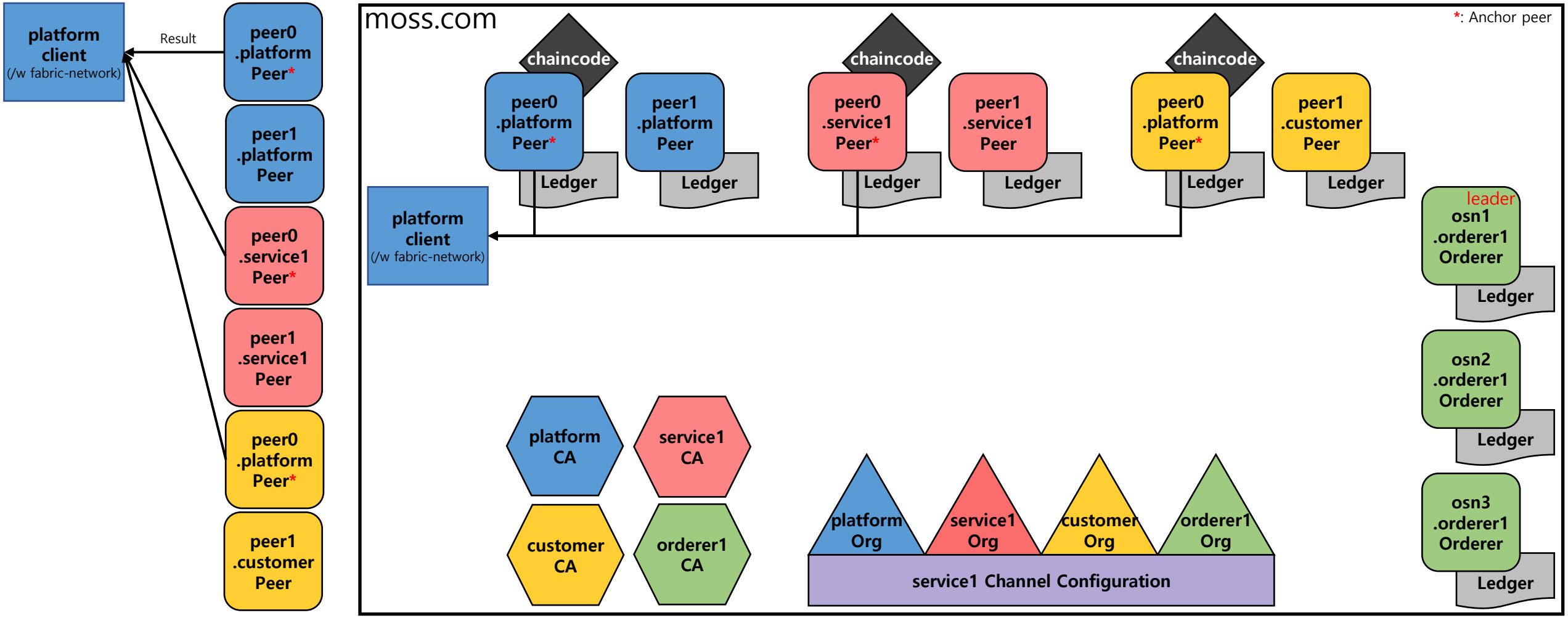
트랜잭션 Flow

5. 생성된 블록을 Peer에 전파(broadcast)
6. 피어는 트랜잭션을 검증(validate)하고, 원장에 커밋(commit)



트랜잭션 Flow

7. Application에 전파

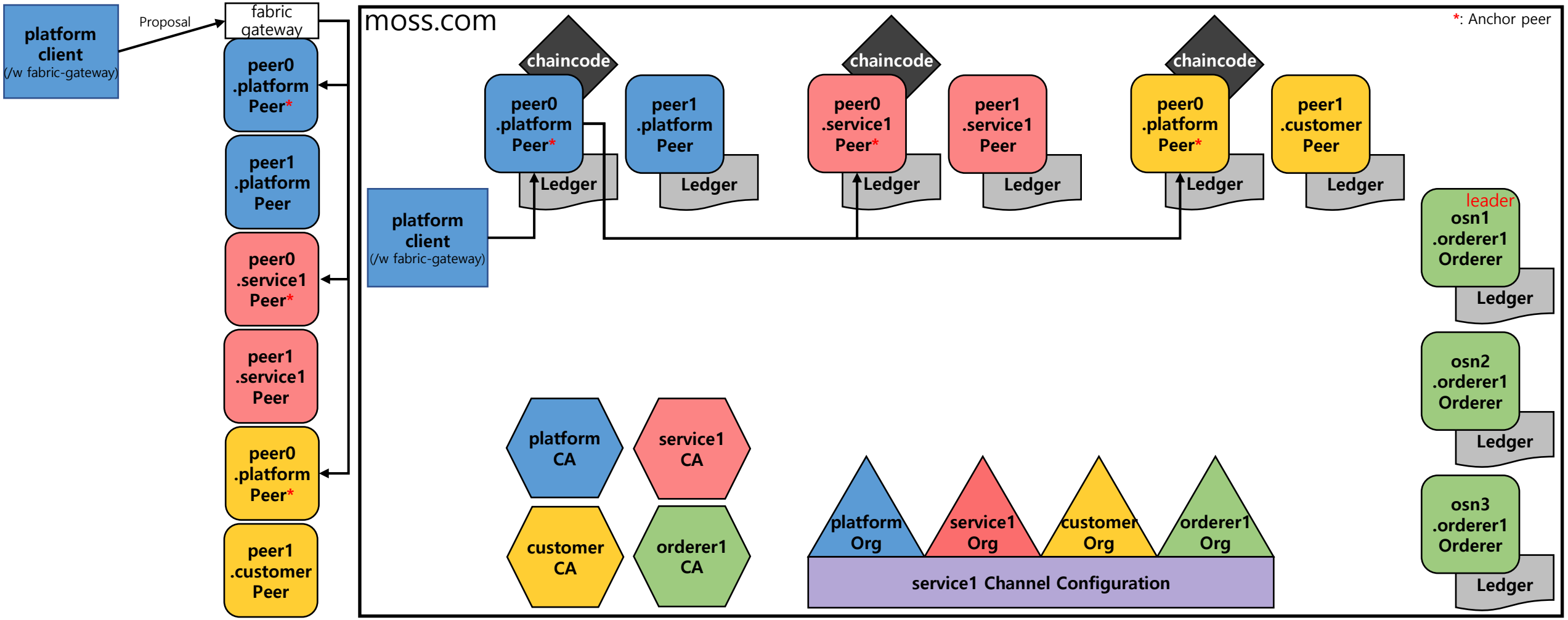


트랜잭션 Flow - fabric-gateway (v2.4)

- 기존 방식 단점
  - client가 체인코드가 설치된 모든 peer와 orerder의 접속 정보를 관리
  - 각 peer 및 orderer와 통신 오류를 client에서 처리해야 함 처리
- fabric-gateway
  - Fabric v2.4 부터 peer에 fabric-gateway 기능 추가
  - client(application)은 단일 peer와 통신하여 트랜잭션 제출 가능
  - fabric-gateway SDK 사용
    - Java, node.js, go
    - https://github.com/hyperledger/fabric-gateway

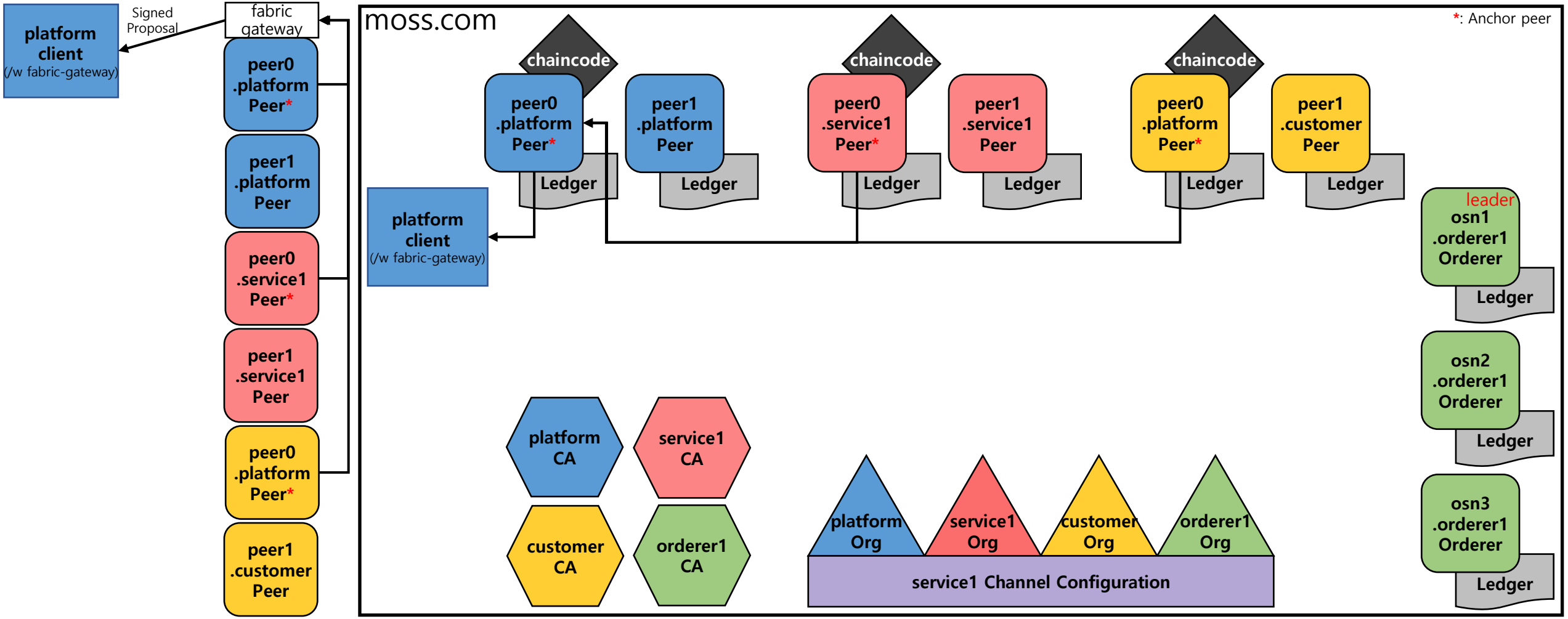
트랜잭션 Flow - fabric-gateway (v2.4)

1-1. 트랜잭션 Proposal to Peer's fabric gateway
1-2. 트랜잭션 Proposal



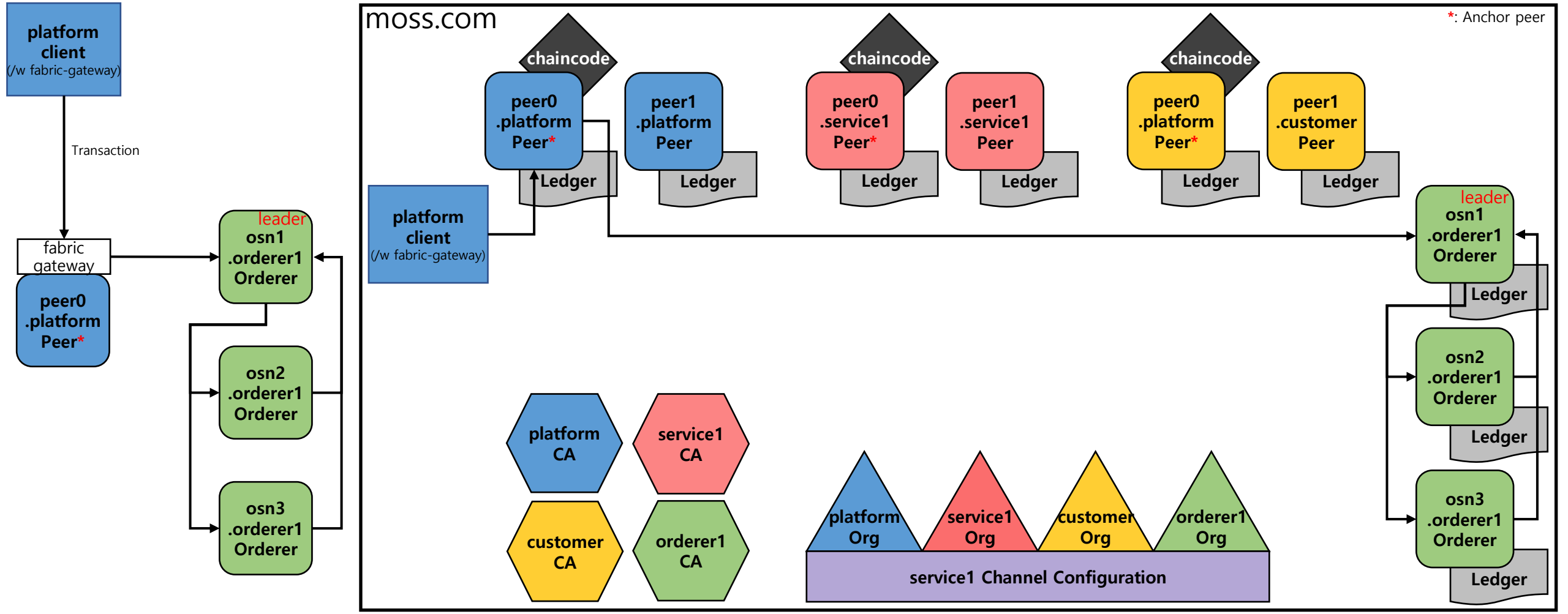
트랜잭션 Flow - fabric-gateway (v2.4)

2. 트랜잭션 Endorsement by Peers (트랜잭션 실행, 검증 및 보증)



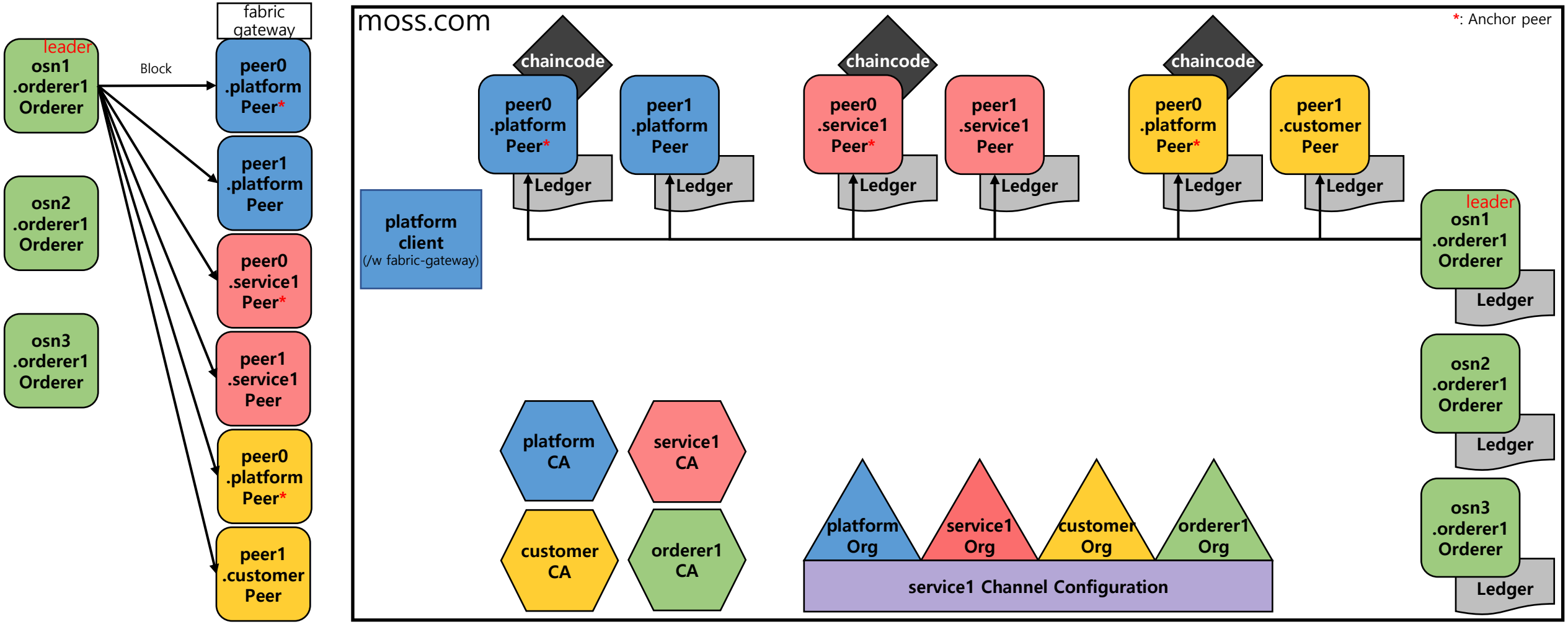
트랜잭션 Flow

3. 트랜잭션을 Ordering 서비스에 제출(submit) (orderer가 leader가 아니면 leader에게 forward)
4. Ordering 서비스는 합의 알고리즘에 따라 트랜잭션 정렬 및 블록 생성



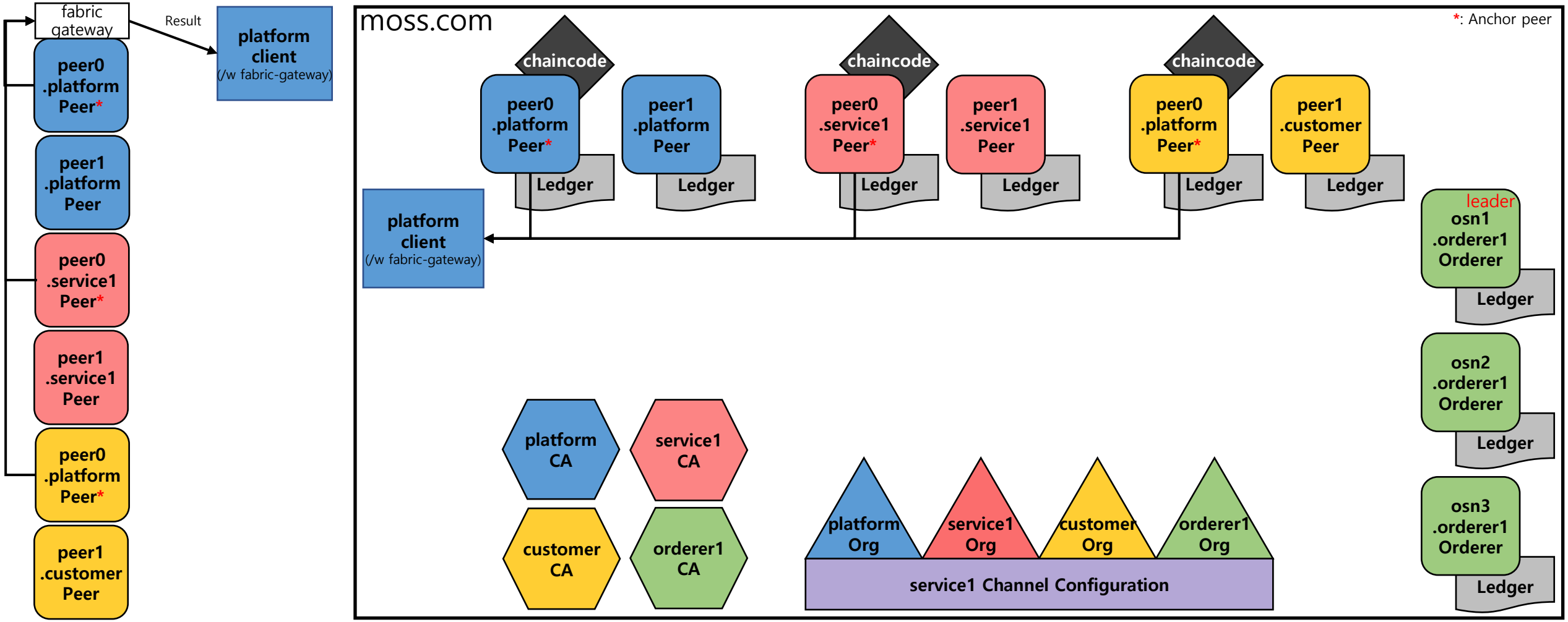
트랜잭션 Flow - fabric-gateway (v2.4)

5. 생성된 블록을 Peer에 전파(broadcast)
6. 피어는 트랜잭션을 검증(validate)하고, 원장에 커밋(commit)



트랜잭션 Flow - fabric-gateway (v2.4)

7. Fabric gateway에 전파
8. Application에 전파



트랜잭션 Flow – 참고

- Hyperledger Fabric - Transaction Flow – 링크

애플리케이션

fabric-gateway Java SDK를 이용하여
platform 조직의 user1 인증정보를 이용해서 peer0.platform.moss.com gateway와 통신
SDK: https://github.com/hyperledger/fabric-gateway

0. Identity 파일
1. 변수설정 (mspId, channelName, chaincodeName 등)
2. gRPC connection 생성
3. Gateway 생성
4. Network 및 Contract 조회
5. Contract 호출

Source Code

애플리케이션 - 0. Identity 파일

Peer와 통신을 위한 TLS 인증서 및 User의 CA 인증서, 개인키 필요

1. Peer peer0.platform.moss.com TLS 인증서

./.../peers/peer0.platform.moss.com/tls/ca.crt

2. User User1@platform.moss.com CA 인증서

./.../users/User1@platform.moss.com/msp/keystore/c676ec3f2255542060705ec553467b0765c68a161148f736db3cf39687278e46_sk

3. User User1@platform.moss.com CA 개인키

./.../users/User1@platform.moss.com/msp/signcerts/cert.pem

애플리케이션 - 1. 변수설정

통신에 필요한 변수 설정

- MspId
- ChannelName
- ChaincodeName
- Peer TLS Certificates
- User CA Certificate
- User CA PrivateKey
- Peer Endpoint
- Peer HostAlias

```
private static final String mspId = "platformMSP";
private static final String channelName = "service1";
private static final String chaincodeName = "simple";

// Path to crypto materials.
private static final Path cryptoPath = Paths.get(System.getProperty("user.dir"), "…", "peerOrganizations", "platform.moss.com");
// Path to user certificate.
private static final Path certPath = cryptoPath.resolve(Paths.get("users", "User1@platform.moss.com", "msp", "signcerts", "cert.pem"));
// Path to user private key directory.
private static final Path keyDirectoryPath = cryptoPath.resolve(Paths.get("users", "User1@platform.moss.com", "msp", "keystore"));
// Path to peer tls certificate.
private static final Path tlsCertPath = cryptoPath.resolve(Paths.get("peers", "peer0.platform.moss.com", "tls", "ca.crt"));

// Gateway peer end point.
private static final String peerEndpoint = "localhost:8060";
private static final String peerHostAlias = "peer0.platform.moss.com";
```

애플리케이션 - 2. gRPC connection 생성

gRPC connection 생성
동일 endpoint를 사용하는 모든 gateway에서 공유.

```
// Path to peer tls certificate.
private static final Path tlsCertPath =
    cryptoPath.resolve(Paths.get("peers", "peer0.platform.moss.com", "tls", "ca.crt"));

// Gateway peer end point.
private static final String peerEndpoint = "localhost:8060";
private static final String peerHostAlias = "peer0.platform.moss.com";

private static ManagedChannel newGrpcConnection() throws IOException, CertificateException {
    Reader tlsCertReader = Files.newBufferedReader(tlsCertPath);
    X509Certificate tlsCert = Identities.readX509Certificate(tlsCertReader);

    return NettyChannelBuilder.forTarget(peerEndpoint)
            .sslContext(GrpcSslContexts.forClient().trustManager(tlsCert).build())
            .overrideAuthority(peerHostAlias)
            .build();
}
```

애플리케이션 - 3. Gateway 생성

gRPC connection 및 인증정보로 Gateway 생성

```
public static void main(String[] args) throws Exception {
    ManagedChannel channel = newGrpcConnection();

    // gateway 커넥션 생성
    Gateway gateway = Gateway.newInstance()
            .connection(channel)  // gRPC channel
            .identity(newIdentity())  // new X509Identity(mspId, userCertificate)
            .signer(newSigner())  // Signers.newPrivateKeySigner(privateKey);
            // Default timeouts for different gRPC calls
            .evaluateOptions(CallOption.deadlineAfter(5, TimeUnit.SECONDS))
            .endorseOptions(CallOption.deadlineAfter(15, TimeUnit.SECONDS))
            .submitOptions(CallOption.deadlineAfter(5, TimeUnit.SECONDS))
            .commitStatusOptions(CallOption.deadlineAfter(1, TimeUnit.MINUTES))
            .connect();
}

private static Identity newIdentity() throws IOException, CertificateException {
    Reader certReader = Files.newBufferedReader(certPath);
    X509Certificate certificate = Identities.readX509Certificate(certReader);
    return new X509Identity(mspId, certificate);
}

private static Signer newSigner() throws IOException, InvalidKeyException {
    Path keyPath = Files.list(keyDirectoryPath)
            .findFirst()
            .orElseThrow();
    Reader keyReader = Files.newBufferedReader(keyPath);
    PrivateKey privateKey = Identities.readPrivateKey(keyReader);
    return Signers.newPrivateKeySigner(privateKey);
}
```

애플리케이션 - 4. Network 및 Contract 조회

Channel 이름으로 Network 조회, Chaincode 이름으로 Contract 조회

```
// 네트워크 조회
Network network = gateway.getNetwork(channelName);

// 컨트랙트 조회
Contract contract = network.getContract(chaincodeName);
```

Gateway.getNetwork() – Channel == Network

```
public interface Gateway extends AutoCloseable {
    /**
     * Returns an object representing a network.
     *
     * @param networkName The name of the network (channel name)
     * @return A network.
     * @throws NullPointerException if the network name is null.
     */
    Network getNetwork(String networkName);
}
```

Network.getContract() – contractName이 없으면 @Default Contract 호출

```
public interface Network {
    Contract getContract(String chaincodeName);
    Contract getContract(String chaincodeName, String contractName);
}
```

애플리케이션 - 5. Contract 호출

Contract 호출

```
// get
value = get(contract);

// set
String newValue = "newValue: " + Instant.now().getEpochSecond();
set(contract, newValue);

// get
value = get(contract);
```

```
/**
 * Evaluate: 조회
 */
private static String get(Contract contract) throws GatewayException {
    byte[] bytes = contract.evaluateTransaction("get");
    return new String(bytes, StandardCharsets.UTF_8);
}

/**
 * Submit: 수정
 */
private static void set(Contract contract, String value)
        throws CommitStatusException, EndorseException, CommitException, SubmitException {
    contract.submitTransaction("set", value);
}
```

애플리케이션 - 5. Contract 호출

호출시간 측정

```
// get
long startTime = System.currentTimeMillis();
String value = get(contract);
long endTime = System.currentTimeMillis();
print("get - value: " + value + " [" + (endTime - startTime) + "ms]");

// set
String newValue = "newValue: " + Instant.now().getEpochSecond(); // newValue: 1647512835
startTime = System.currentTimeMillis();
set(contract, newValue);
endTime = System.currentTimeMillis();
print("set - [" + (endTime - startTime) + "ms]");

// get
startTime = System.currentTimeMillis();
value = get(contract);
endTime = System.currentTimeMillis();
print("get - value: " + value + " [" + (endTime - startTime) + "ms]");
```

```
get - value: newValue [1084ms]
set - [2092ms]
get - value: newValue: 1647512835 [12ms]
```

애플리케이션

Transaction Flow 와 SDK API 비교

```
// 0. Gateway 연결, Network 및 Contract 조회
Gateway gateway = ...;
Network network = gateway.getNetwork("channelName");
Contract contract = network.getContract("chaincodeName", “contractName");

// 1~2. Proposal 및 Endorsement
Proposal proposal = contract.newProposal(“functionName").addArguments("arg1", "arg2").build();
String txId = proposal.getTransactionId();
byte[] digest = proposal.getDigest();
Transaction transaction = proposal.endorse();

// 3~6. Submit
SubmittedTransaction submittedTransaction = transaction.submitAsync();

// 7~8 Ordering, Update Ledger, Result
byte[] resultBytes = submittedTransaction.getResult();
String result = new String(resultBytes, StandardCharsets.UTF_8);
Status status = submittedTransaction.getStatus();
boolean isSuccessful = status.isSuccessful();
long blockNumber = status.getBlockNumber();
TransactionPackage.TxValidationCode txValidationCode = status.getCode();
```

애플리케이션 – 실습

- Hyperledger Fabric Application 개발 – 링크
- 참고
  - [공식문서] Running a Fabric Application
  - fabric-samples, asset-transfer-basic, application 테스트

- END -