

A person is visible from the chest up, wearing a yellow and black horizontally striped shirt. The background is a plain, light gray.

ORACLE



Oracle Blockchain Platform 소개

Wonjo Yoo

Principal Solution Engineer

Cloud Excellence Team, Oracle Korea

Oct 22, 2019



Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

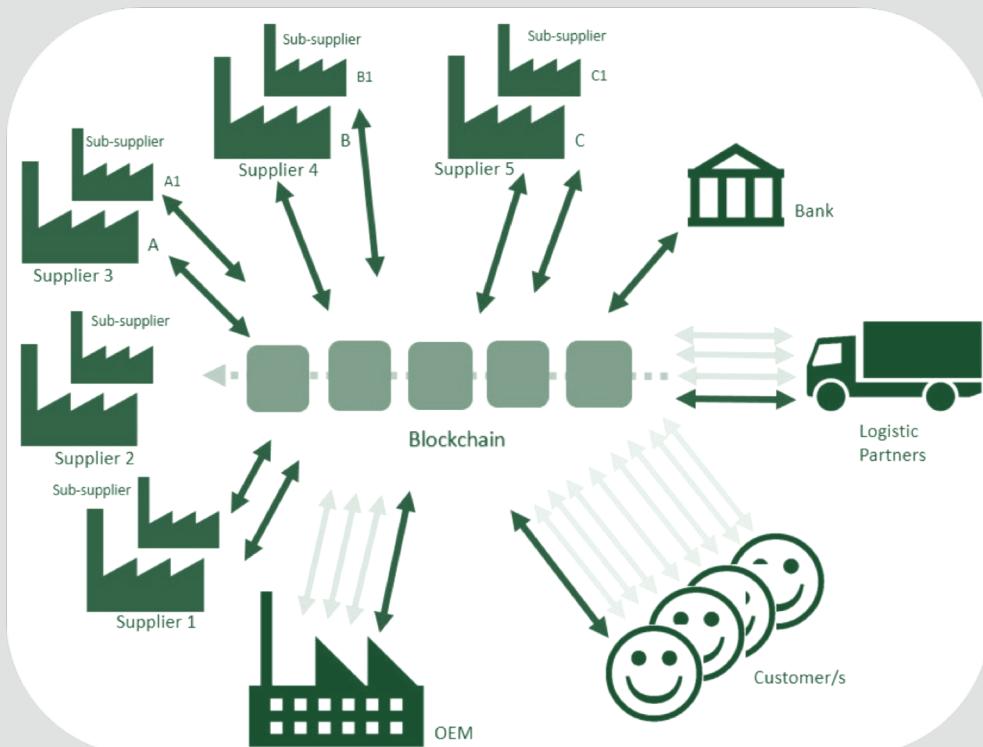


Program agenda

- 1 Hyperledger Fabric
- 2 Oracle Blockchain Platform

Hyperledger Fabric

블록체인의 5가지 특징



- **준 실시간 업데이트, 합의**
- **중개자 대체 : 암호화로 peer간의 신뢰**
- **신뢰할 수 있는 공유와 투명한 데이터 접근**
- **신뢰할 수 있는 거래와 부인방지**
- **history 를 통한 트랜잭션 조작 방지**

블록체인의 두가지 유형

허가형 Permissioned

- 허가된 멤버만이 네트워크에 참여
- 예: 제조업체 및 공급 업체, 화물 운송 업체 및 브로커 네트워크 등
- 보다 안전하고 향상된 관리 - 사전에 알려진 참가자
- **기업형으로 적합함**

비허가 (Open)

- 인터넷에 있는 누구나 가입하여 원장 정보에 액세스 할 수 있음
- 예: Bitcoin, Ethereum, 등
- 덜 안전하고 빈약한 관리 - 사전에 알려지지 않은 참가자



Hyperledger Fabric

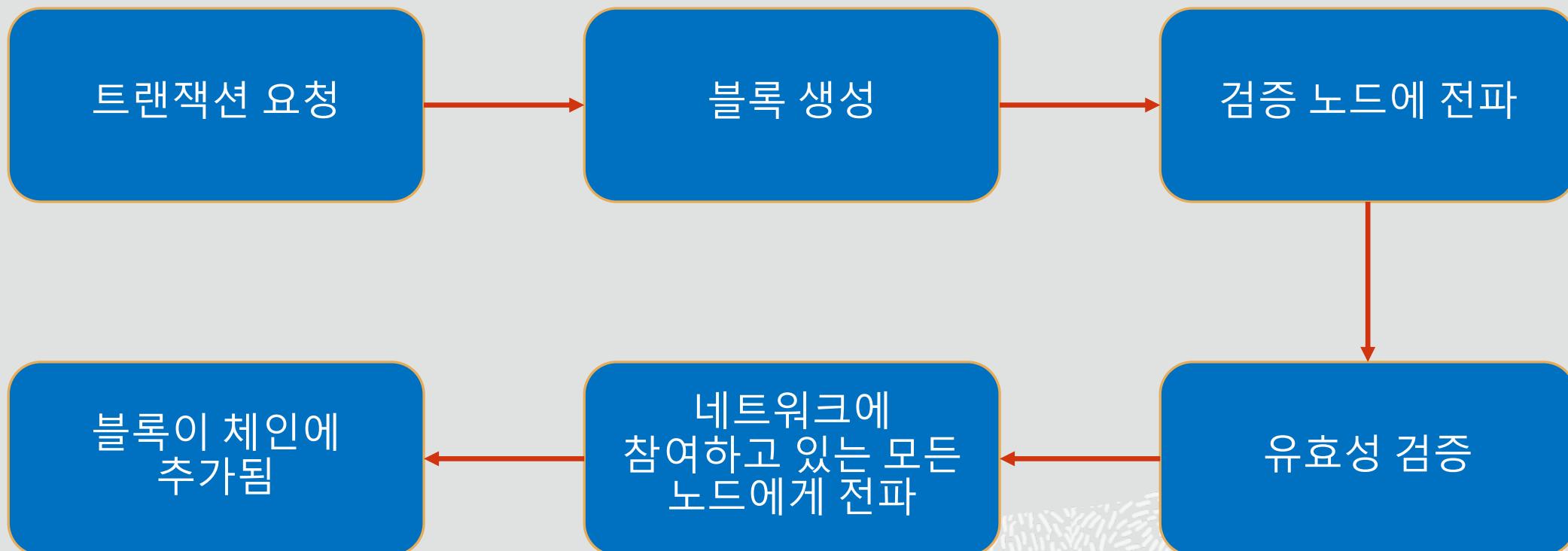
Open source enterprise-grade **permissioned** distributed ledger technology (DLT) platform

<https://www.hyperledger.org/>

<https://hyperledger-fabric.readthedocs.io/en/release-1.4/index.html>

<https://www.youtube.com/watch?v=EKa5Gh9whgU>

기본 동작 방식



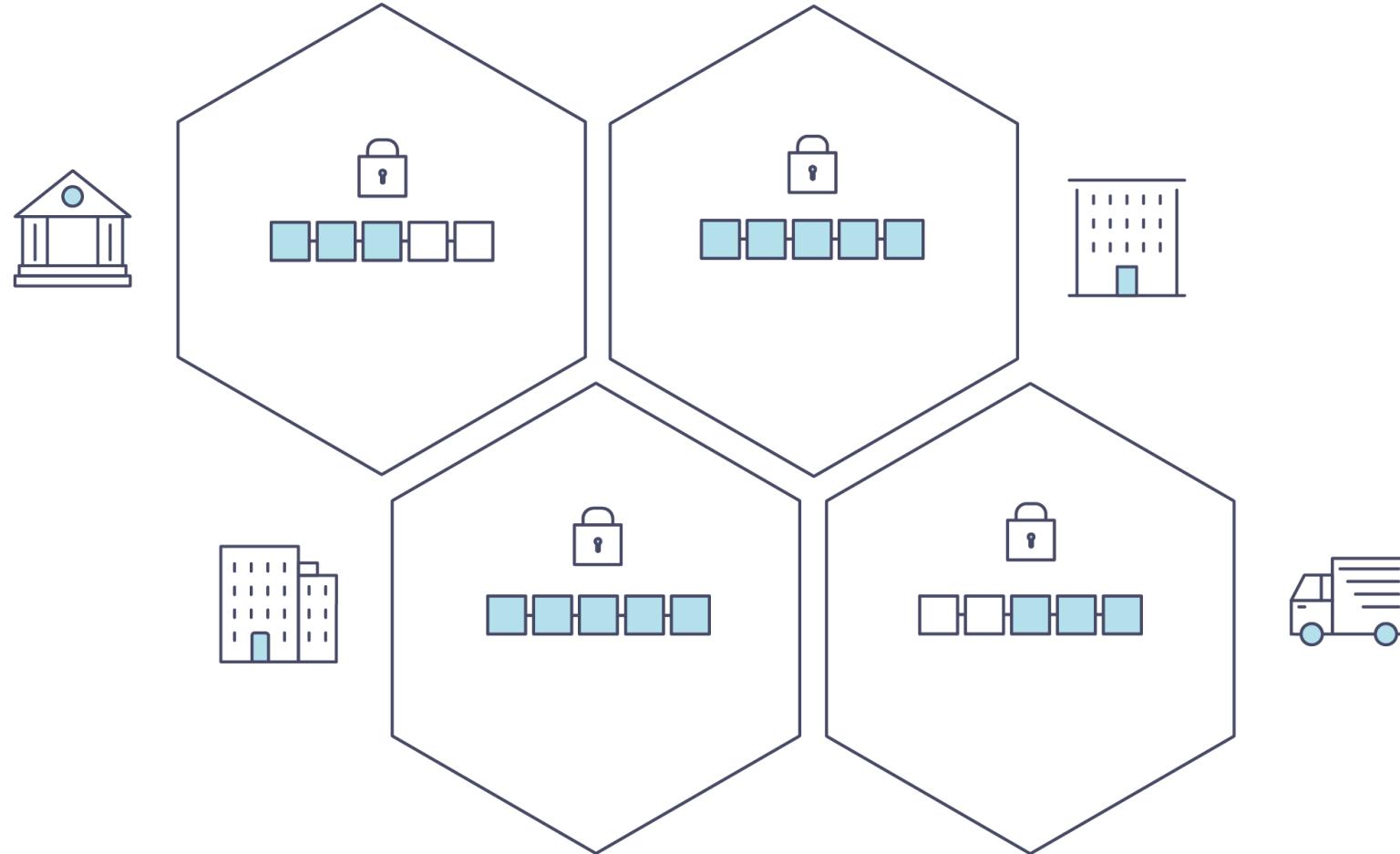
Hyperledger Fabric – Basic Concepts

- Distributed Ledger (분산 원장)
- Smart Contracts (계약서)
- Consensus (합의)



Basic Concepts – Distributed Ledger (분산 원장)

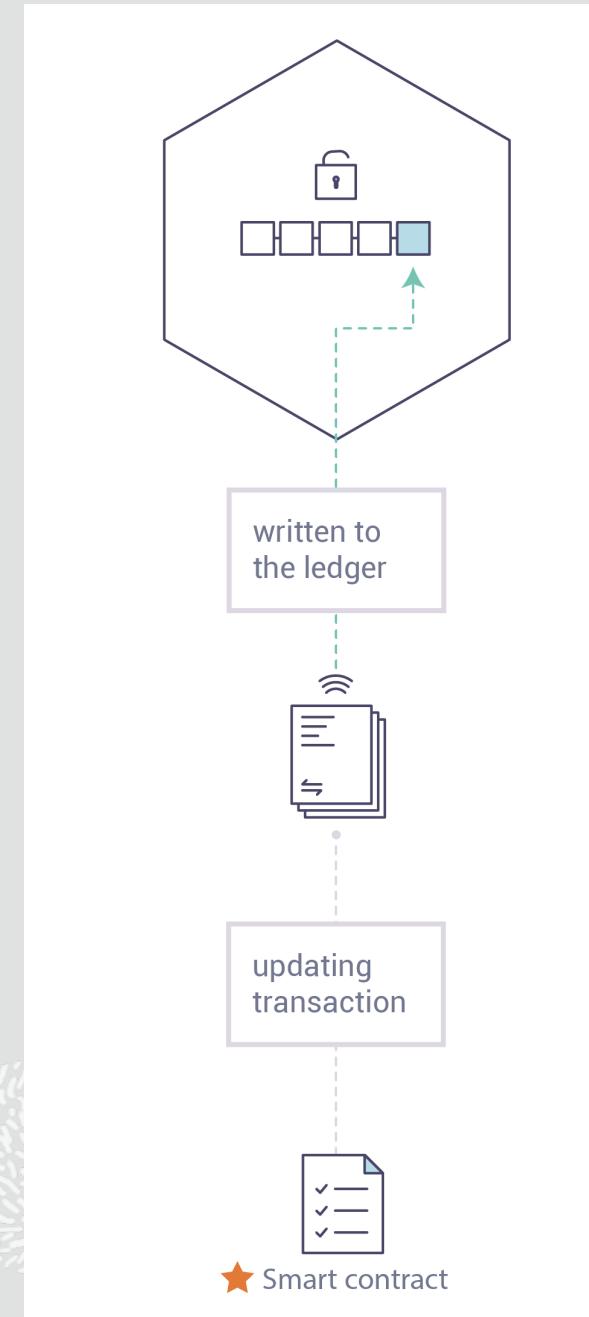
Distributed Ledger



Basic Concepts – Smart Contract (계약서)

Smart Contracts : 계약서

현재 Go, node, java 지원



Basic Concepts – Smart Contract (chaincode)

Seller Organization

ORG1

application:

```
seller = ORG1;  
buyer = ORG2;  
transfer(CAR1, seller, buyer);
```

```
car contract:  
  
query(car):  
    get(car);  
    return car;  
  
transfer(car, buyer, seller):  
    get(car);  
    car.owner = buyer;  
    put(car);  
    return car;  
  
update(car, properties):  
    get(car);  
    car.colour = properties.colour;  
    put(car);  
    return car;
```

Buyer Organization

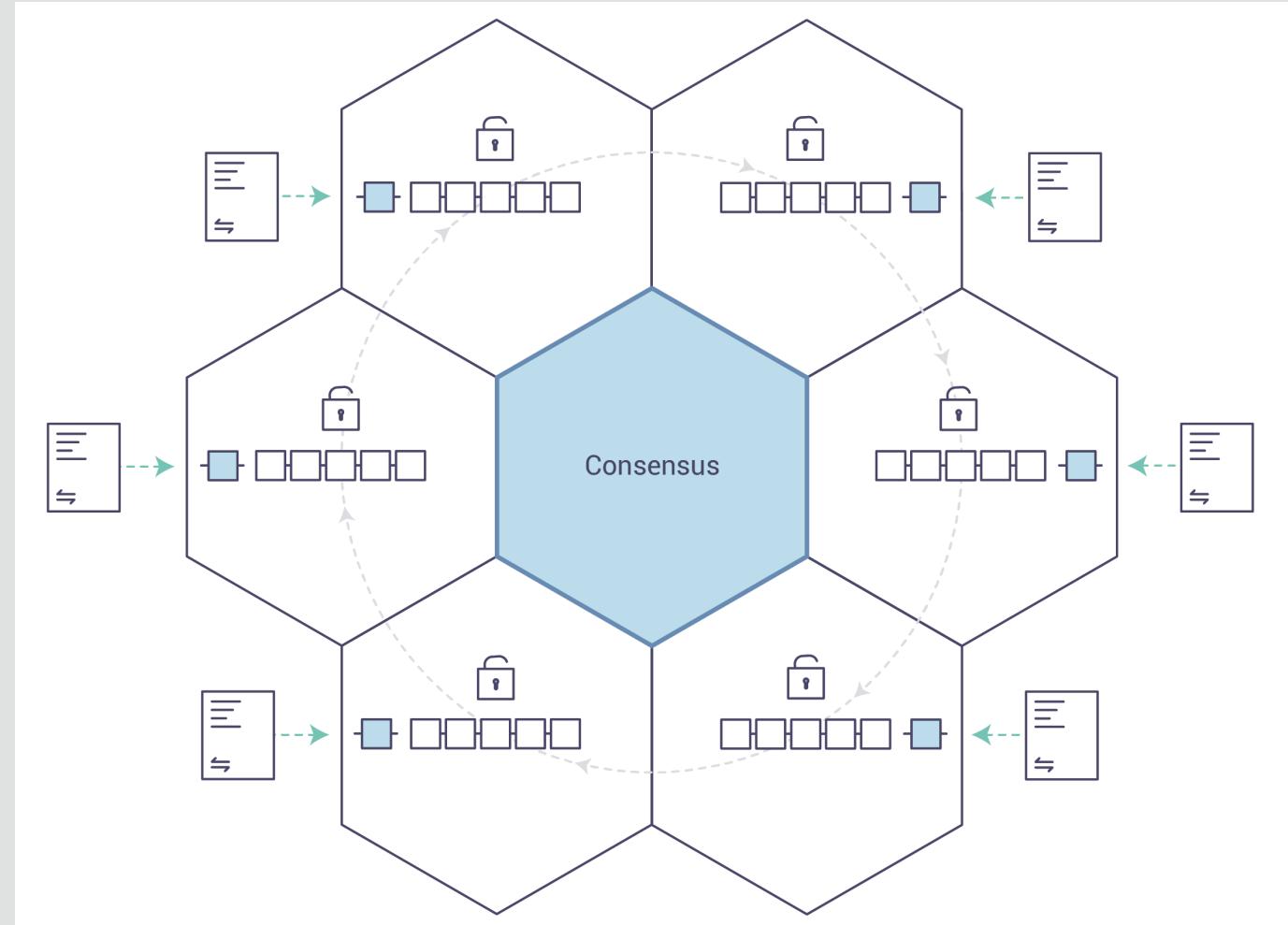
ORG2

application:

```
seller = ORG2;  
buyer = ORG1;  
transfer(CAR2, seller, buyer);
```

Basic Concepts – Consensus(합의)

Consensus



Hyperledger Fabric – Key Concepts

- Peers
- Channel
- Organization
- Ledgers
- Identity
- Membership

Peer의 종류

Peer

트랜잭션을 Commit, Ledger와 WorldState를 보관

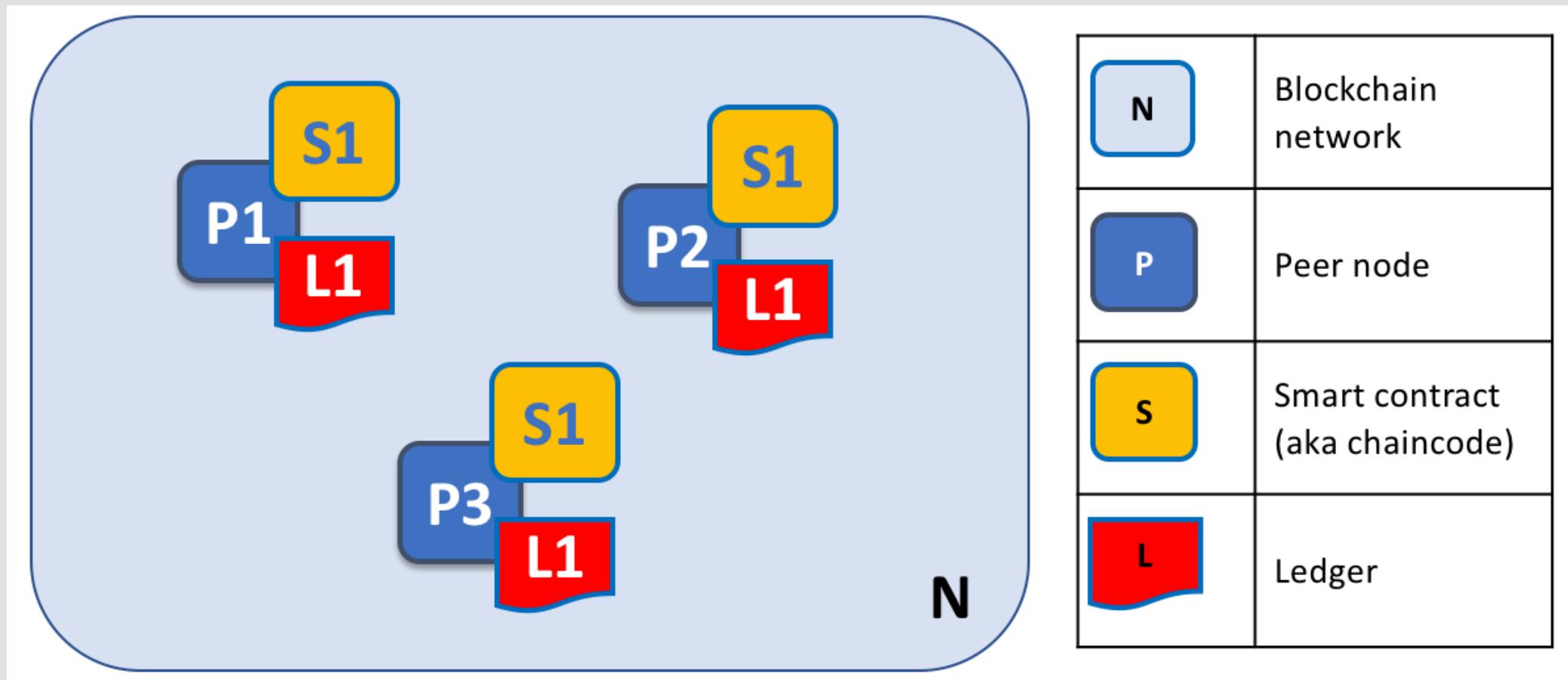
Endorsing Peer

Endorse, Chaincode Execute

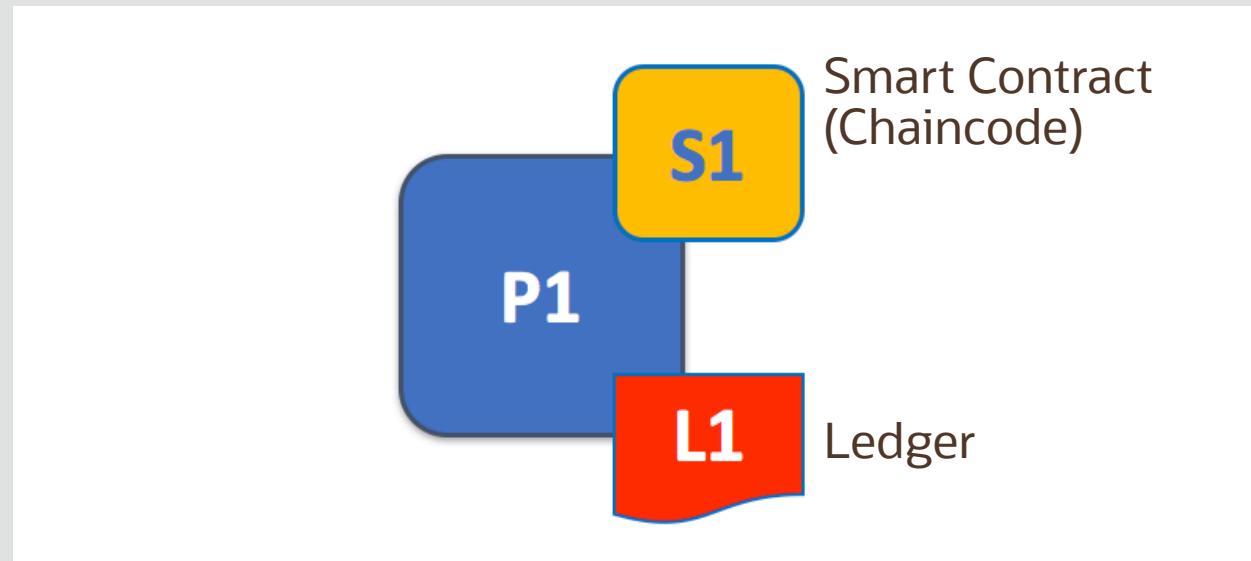
Ordering Peer

TX순서대로 정렬, 블록에 TX추가

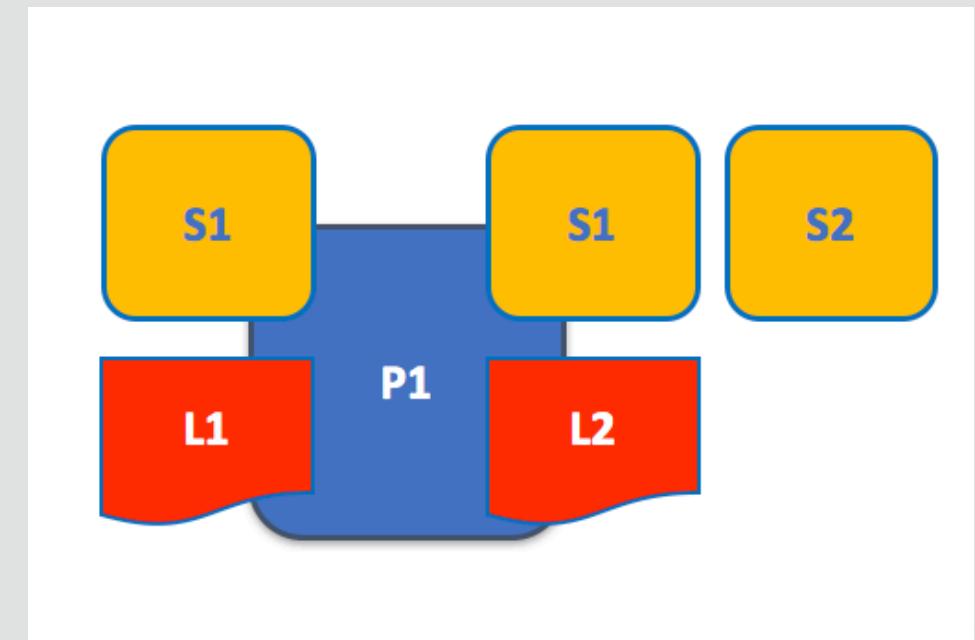
Peer



Peers

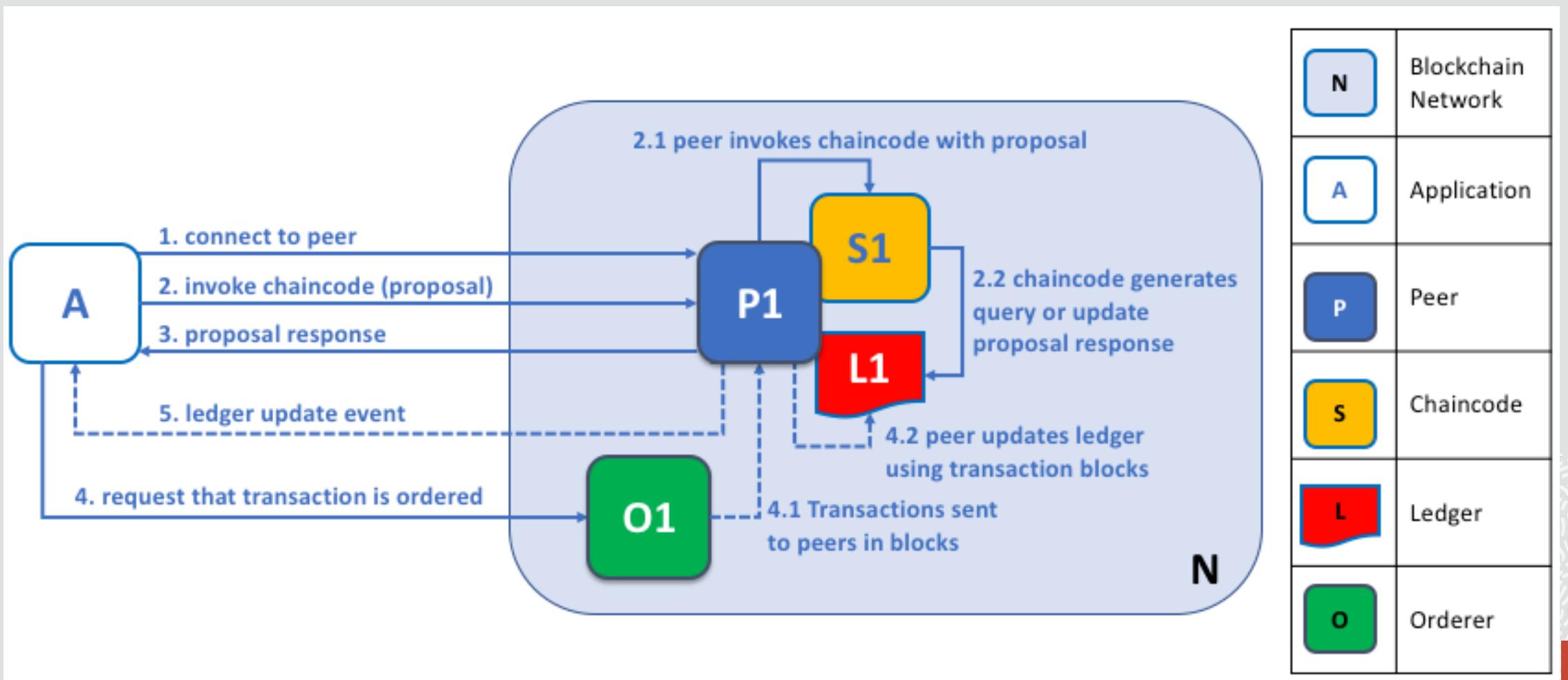


Peer

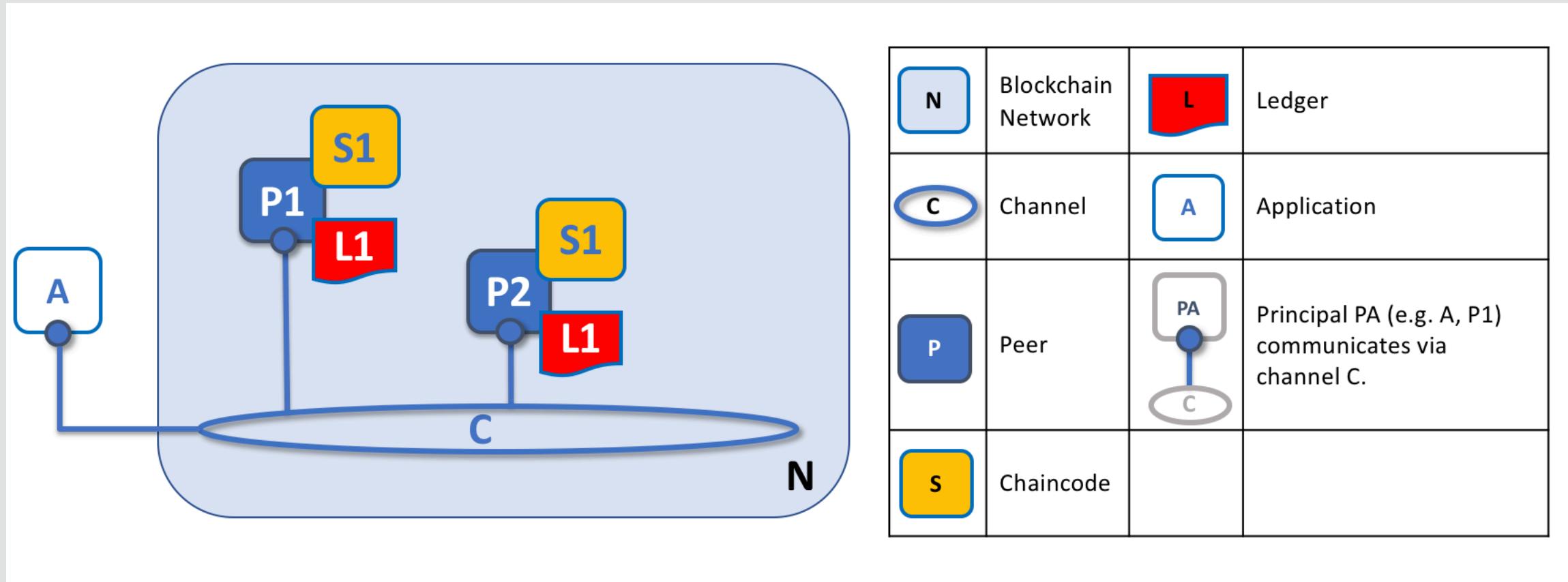


Multiple Ledgers

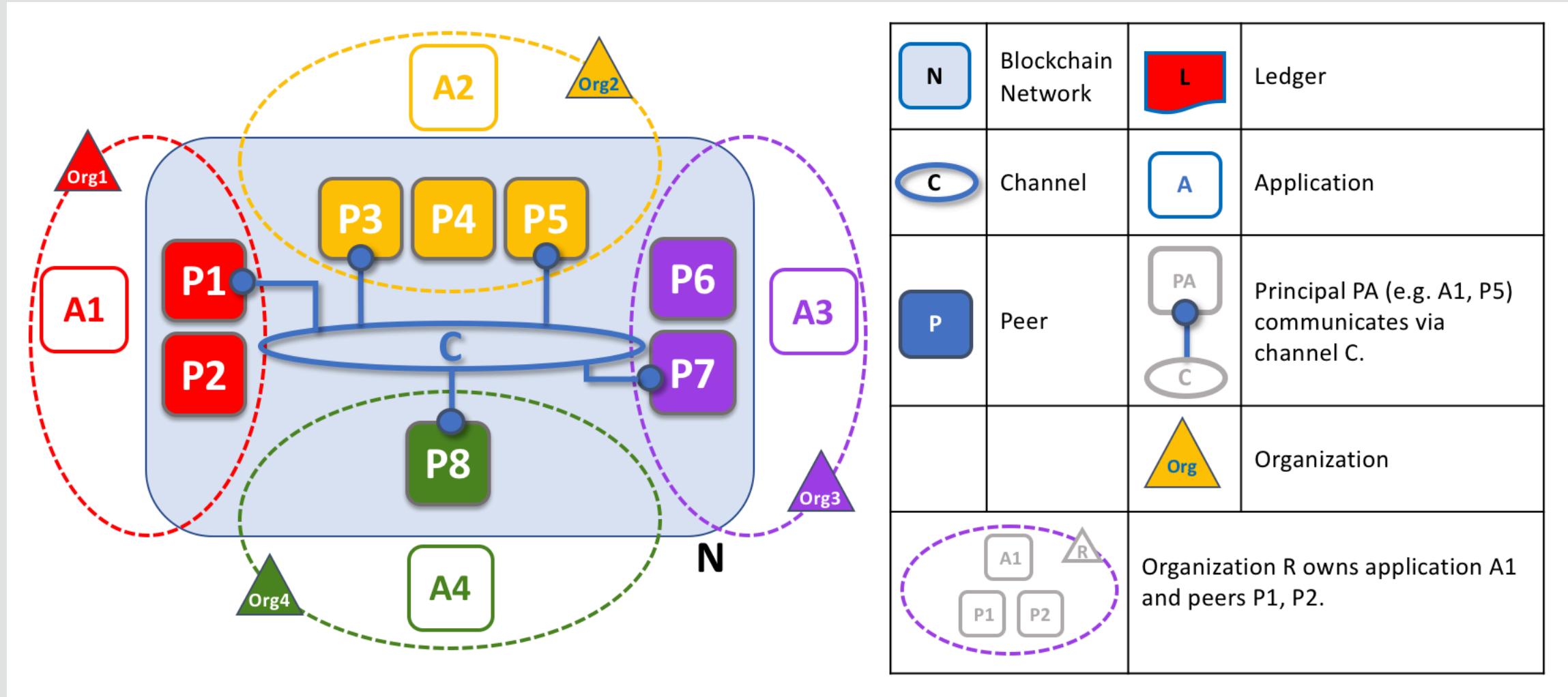
Peers



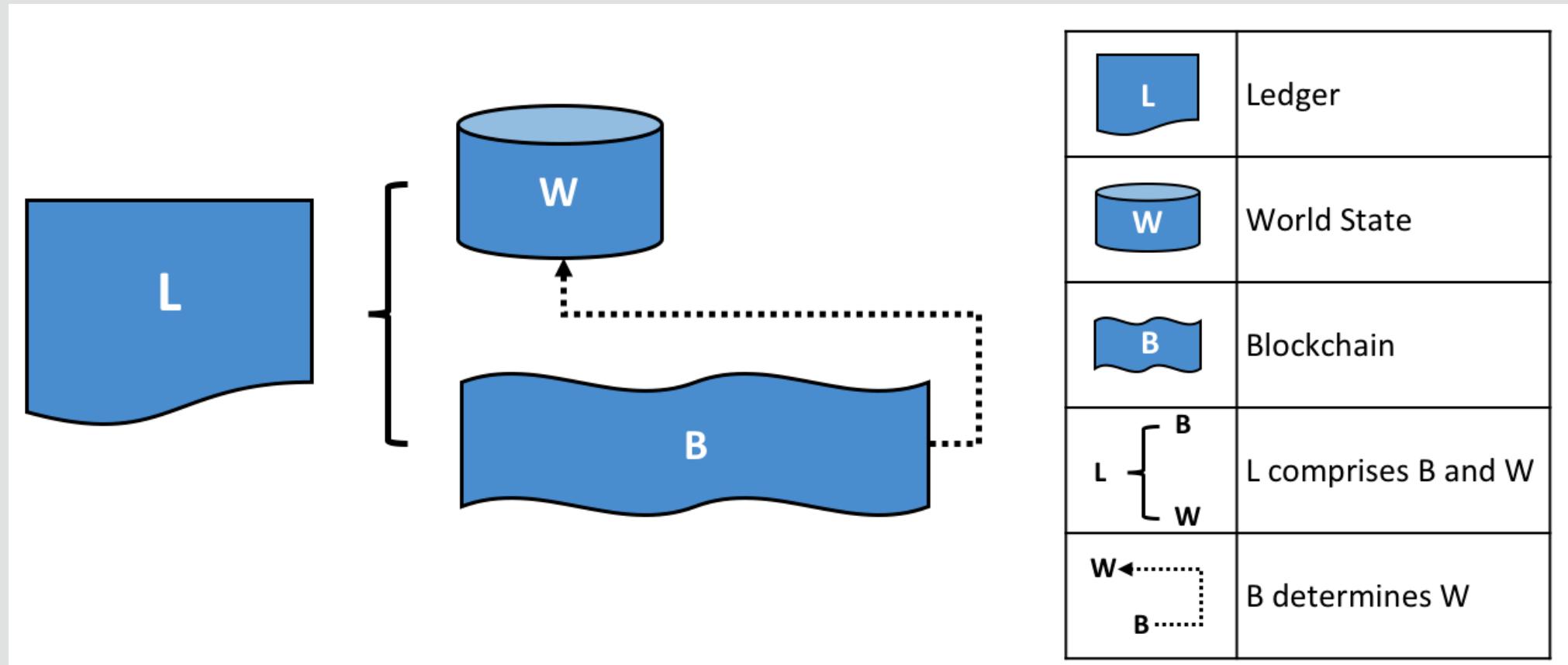
Channel



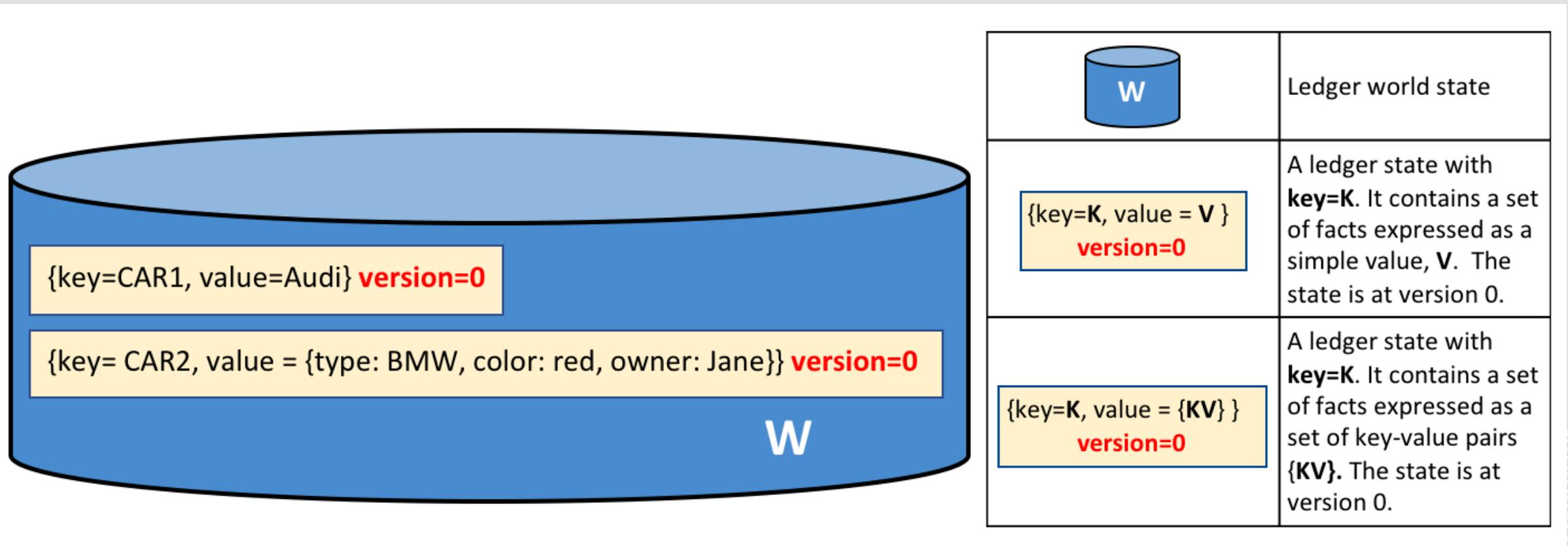
Org(조직)



Ledger



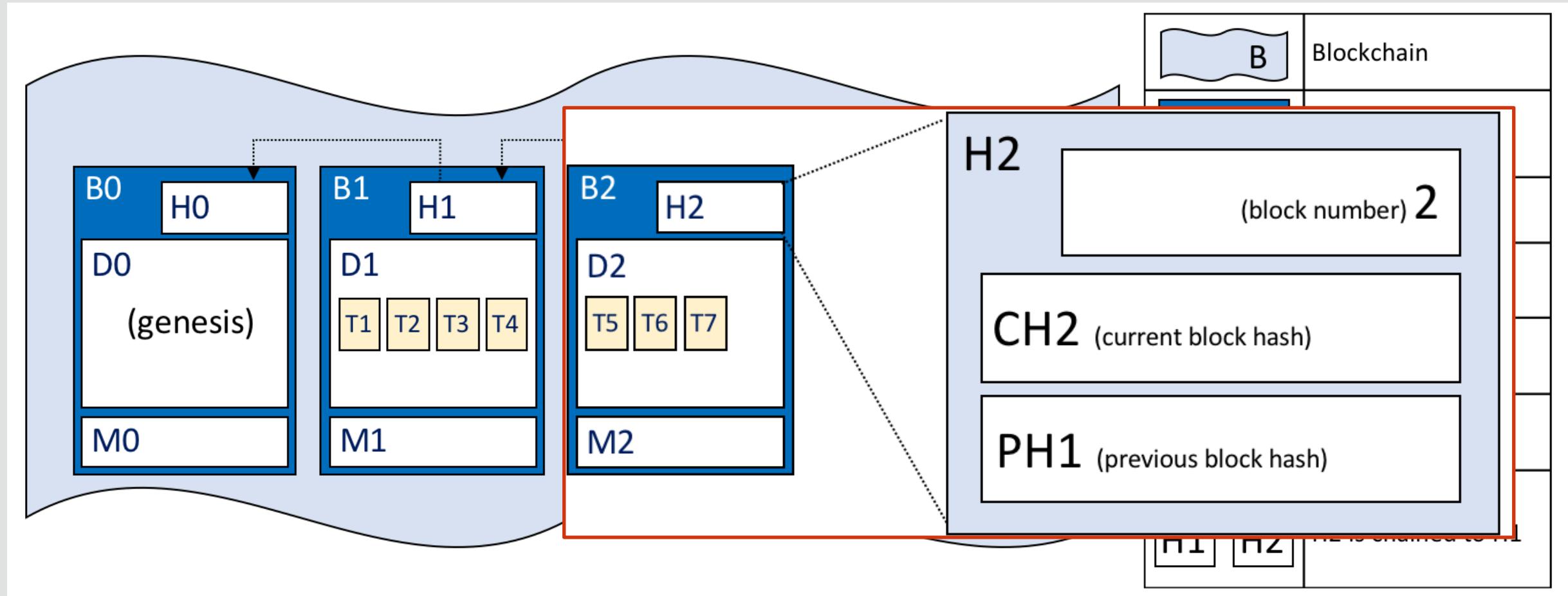
Ledger – world state



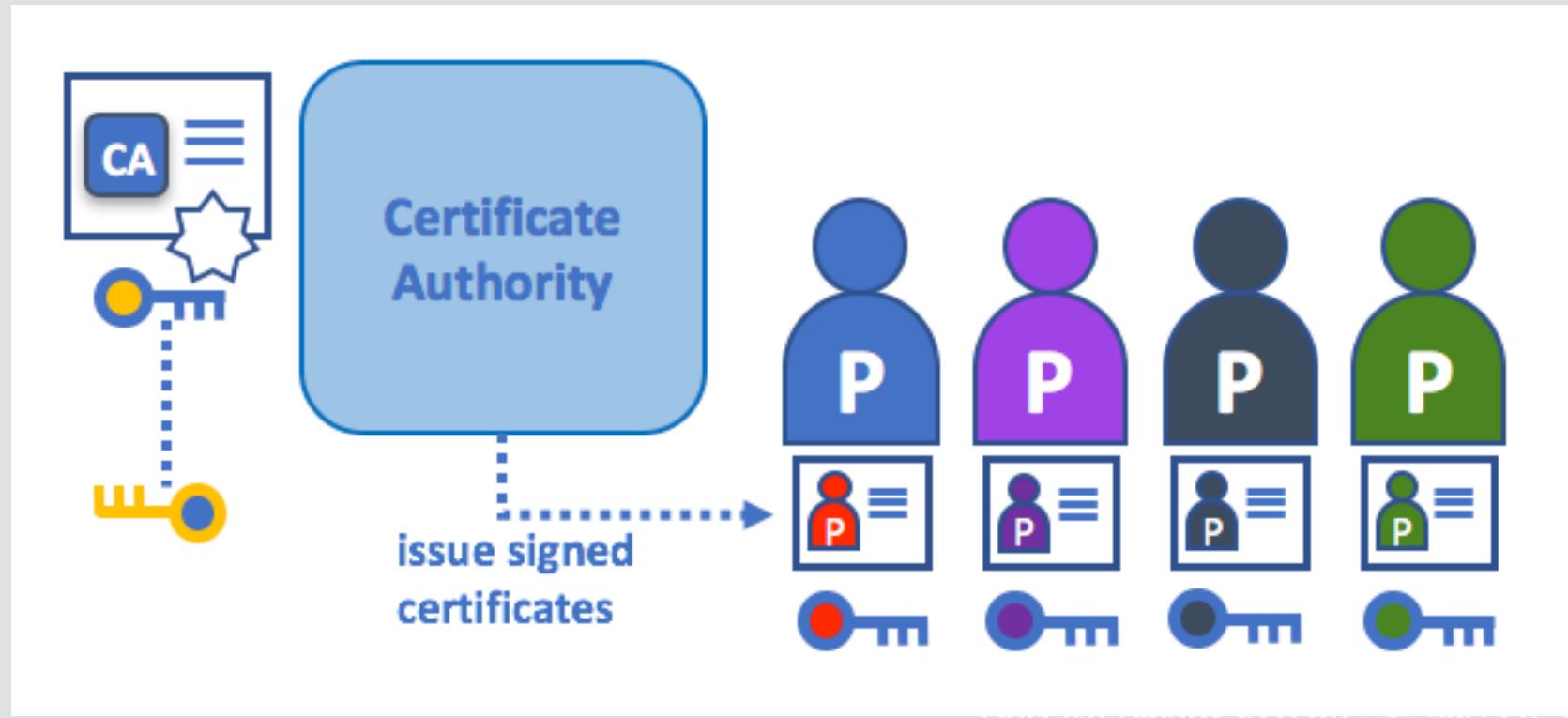
LevelDB and CouchDB.



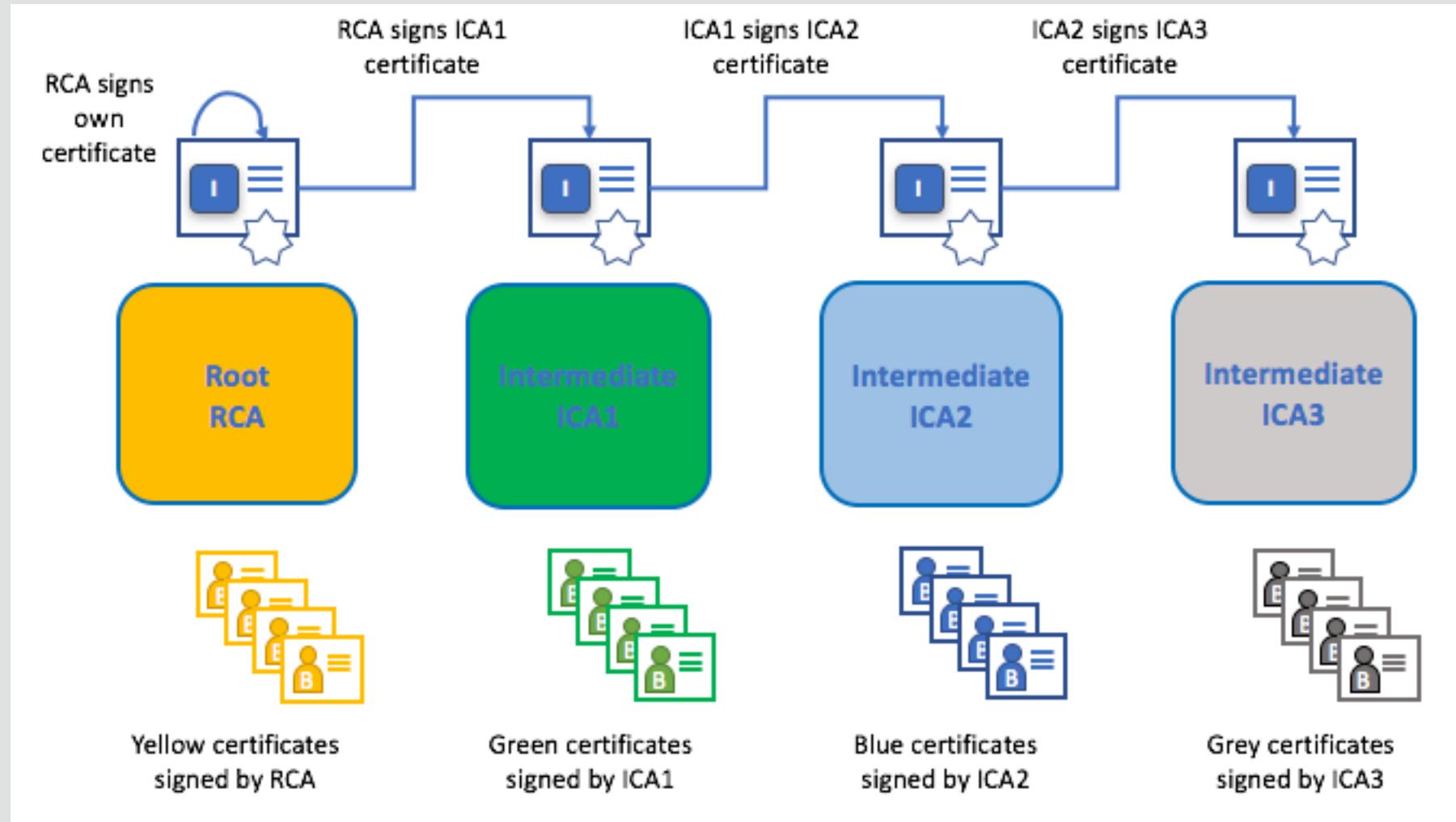
Ledger – blockchain



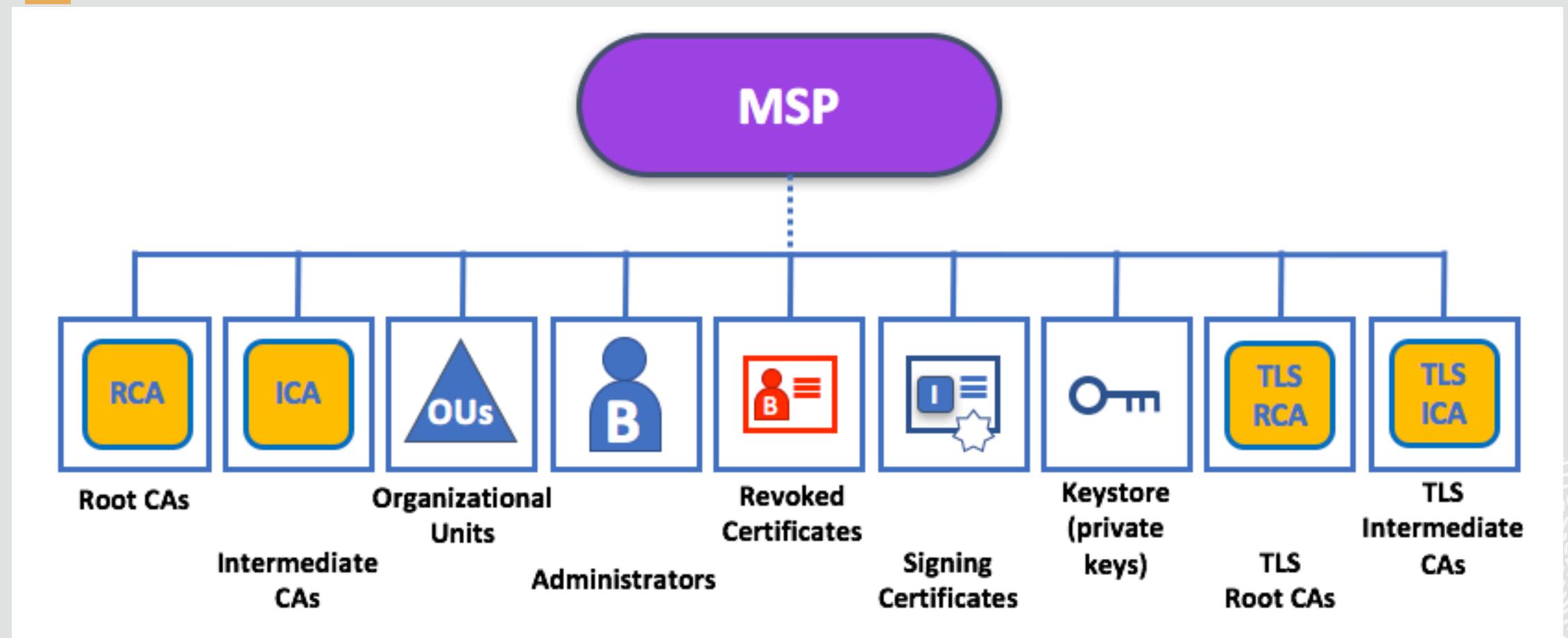
Identity – CA(Certificate Authorities)



Identity – Root CAs, Intermediate CAs and Chains of Trust



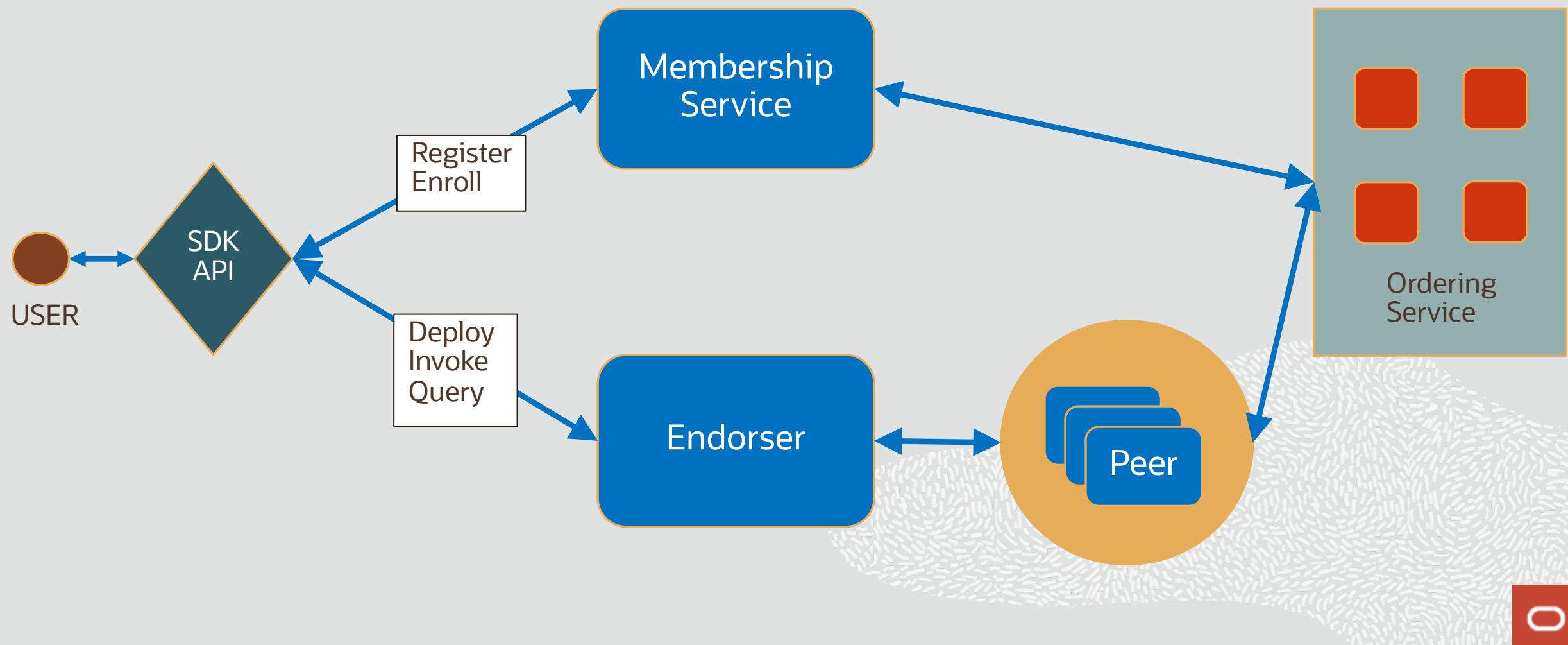
Membership - Membership Service Provider (MSP)



Architectures

Transaction Flows

Basic Architecture



Transaction Flows

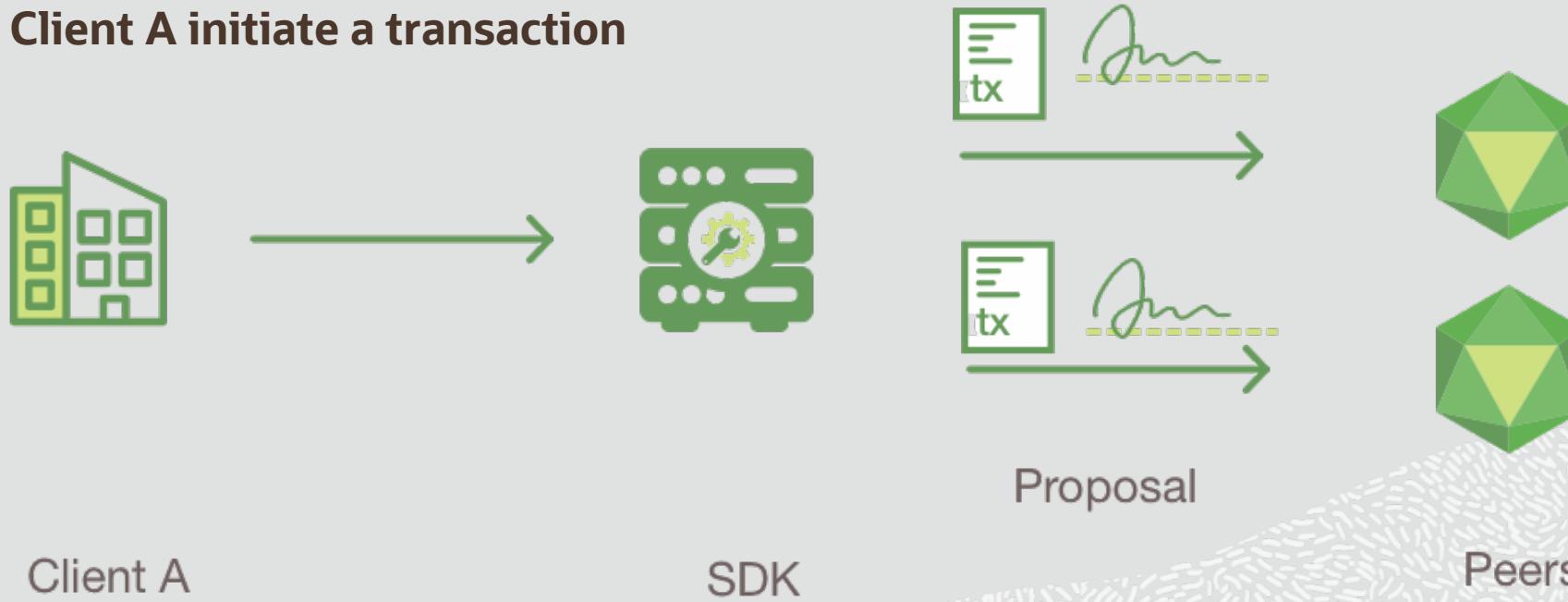
—

시나리오



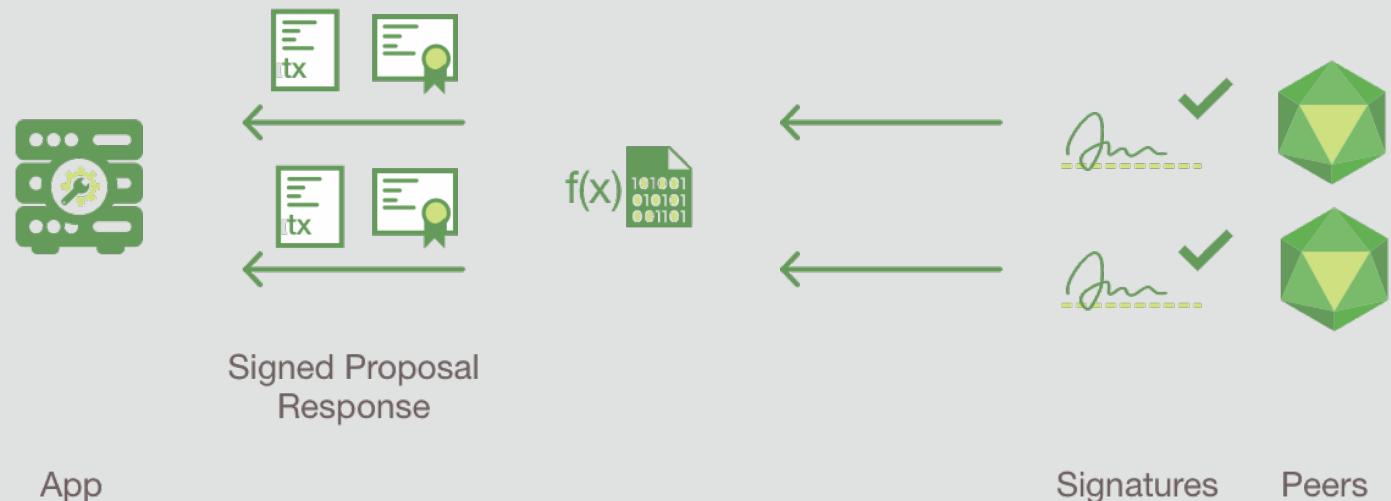
Transaction Flows

1. Propose - Client A initiate a transaction



Transaction Flows

2. Endorsing peers 서명을 검증하고, 트랜잭션을 실행



검증

- (1) Transaction proposal 구조 확인,
- (2) 이전에 실행되었는지 확인 (replay-attack protection),
- (3) 서명이 유효한지 검증 (MSP)
- (4) 권한 확인 (Client A)

실행

- (1) Transaction 실행

응답

- (1) 서명된 응답을 App(SDK)에 Return

Transaction Flows

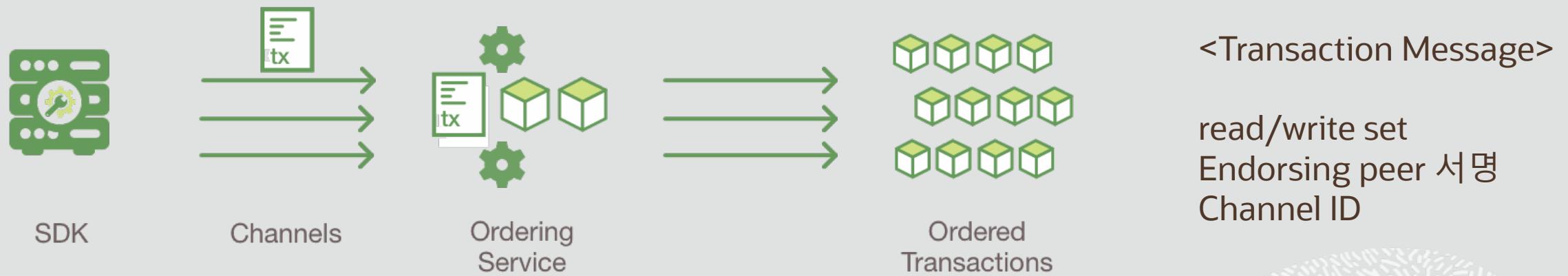
3. Proposal **responses** are inspected



- (1) endorsing peer signatures
- (2) 모든 응답결과가 동일한지

Transaction Flows

4. Ordering - Client assembles endorsements into a transaction



Ordering Service 는 검증을 하지 않고, 채널별로 트랜잭션들을 순서대로 정렬 → 블록생성.

Transaction Flows

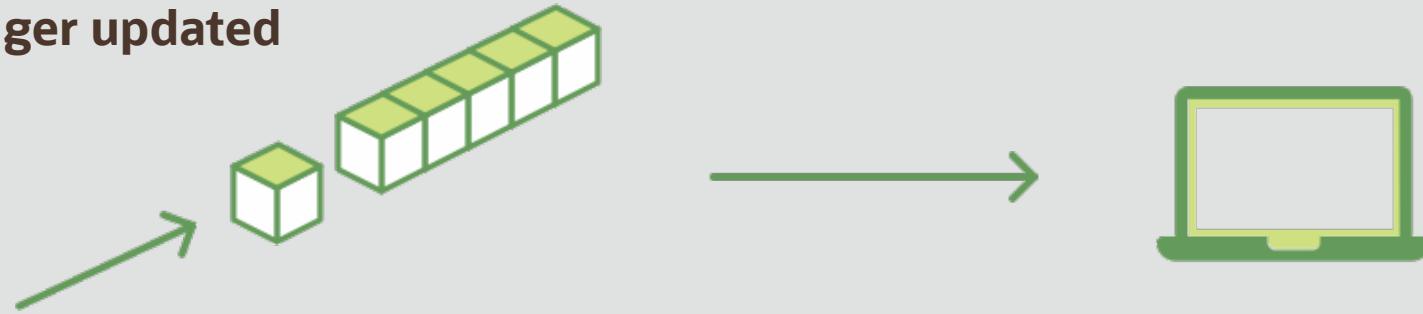
5. Deliver – Transaction을 검증하고 Commit



채널에 있는 모든 Peers에게 블록 전달
Endorsement Policy에 맞는지 검증.
Transaction 수행시의 입력값과 동일한지 확인 후 valid 여부 표시

Transaction Flows

6. Validate - Ledger updated



Appending
Transaction

App

모든 피어에서는 블록을 각 채널의 체인에 추가한다.
Write set을 상태 database에 commit한다.

Hyperledger Fabric의 특징

Open source enterprise-grade permissioned distributed ledger technology (DLT) platform

Modularity : Pluggable(Orderer, membership service provider)

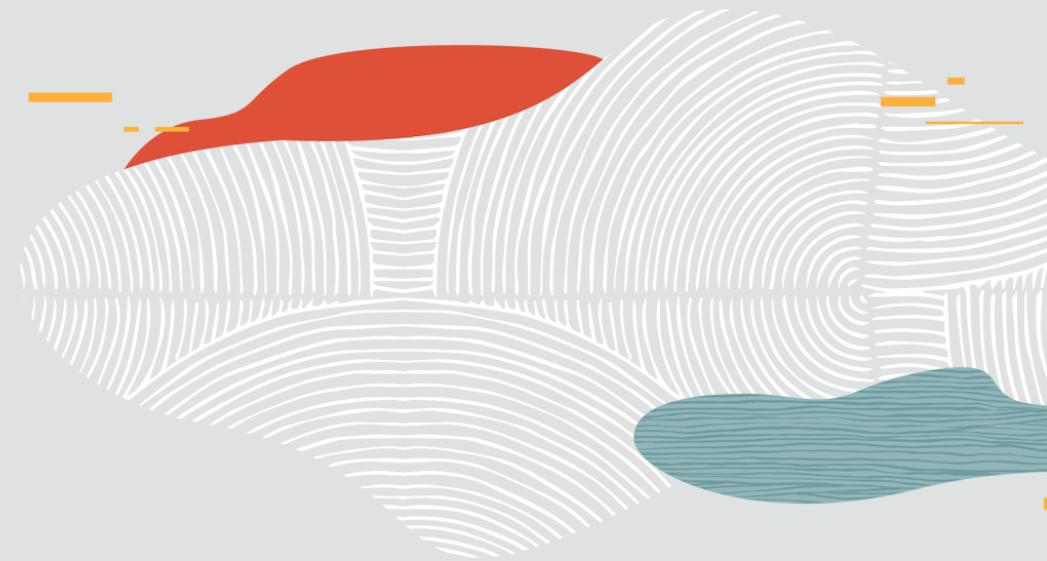
Permissioned Blockchains

Smart Contracts : Java, Go, Node.js

Privacy and Confidentiality

Pluggable Consensus : CFT (crash fault-tolerant) = Kafka+Zookeeper

Performance and Scalability



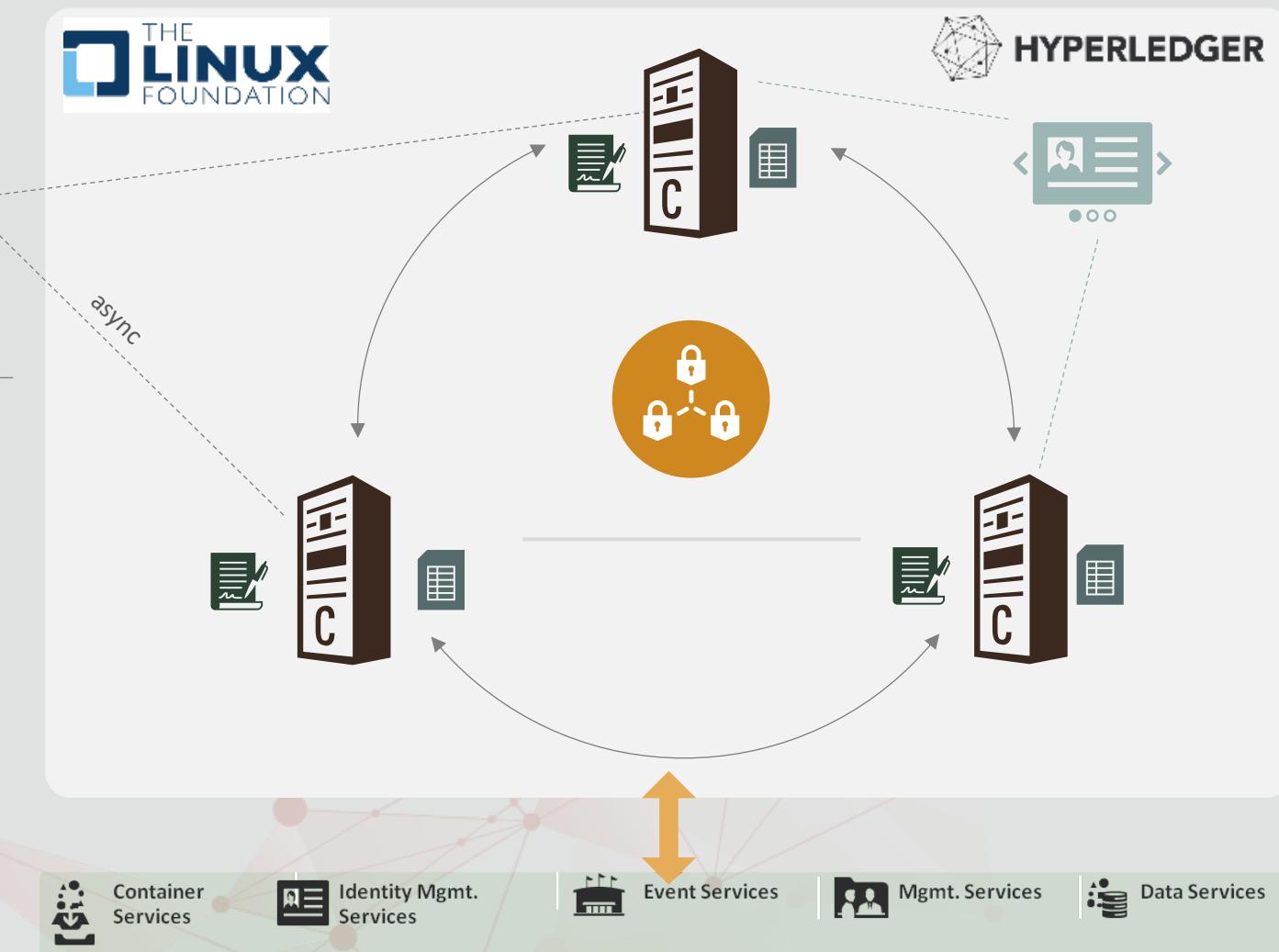
Oracle Blockchain Platform

기업용 블록체인에 필요한 사항

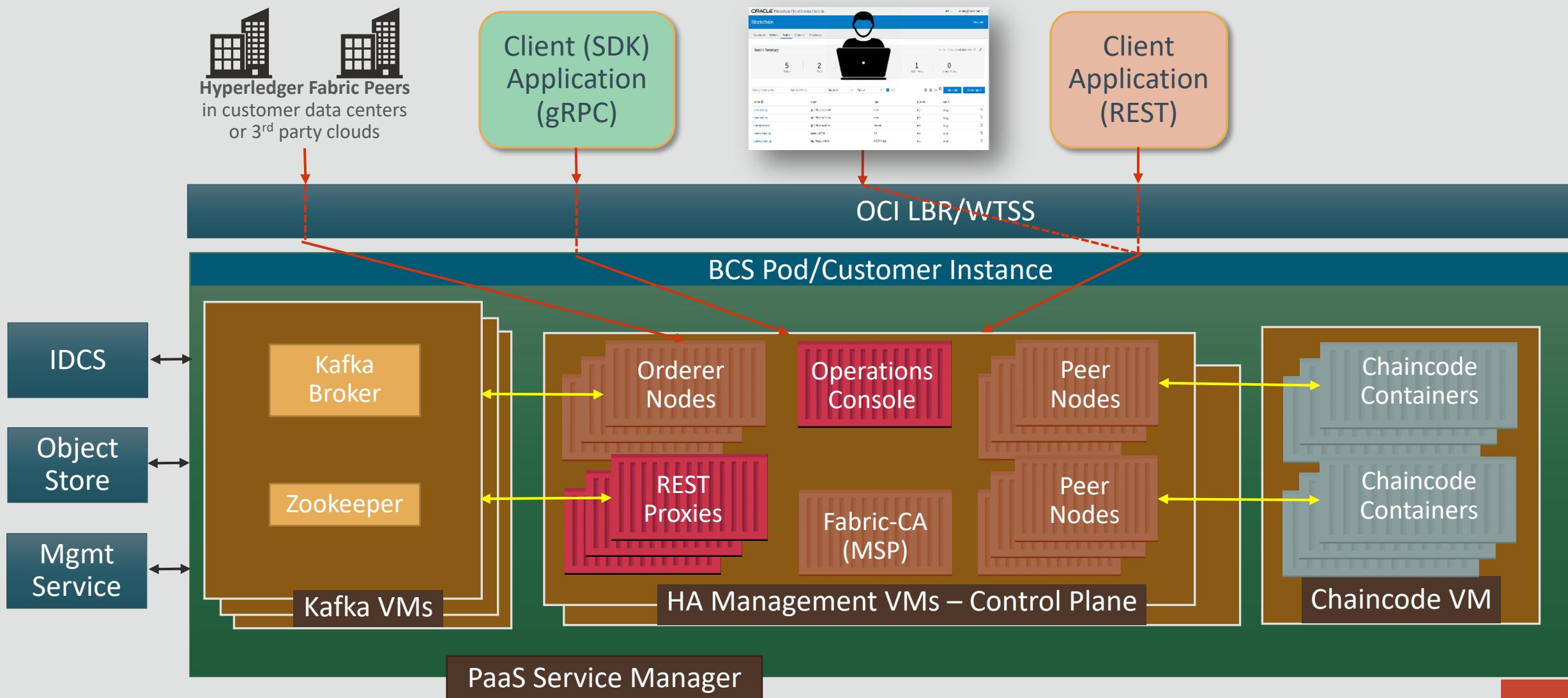
- Validating Nodes/ Peers
- Distributed Ledger
(Single Version of Truth)
- Smart Contract
(aka Chaincode)
- Ordering Service
- Membership Service

추가로 필요한 사항들

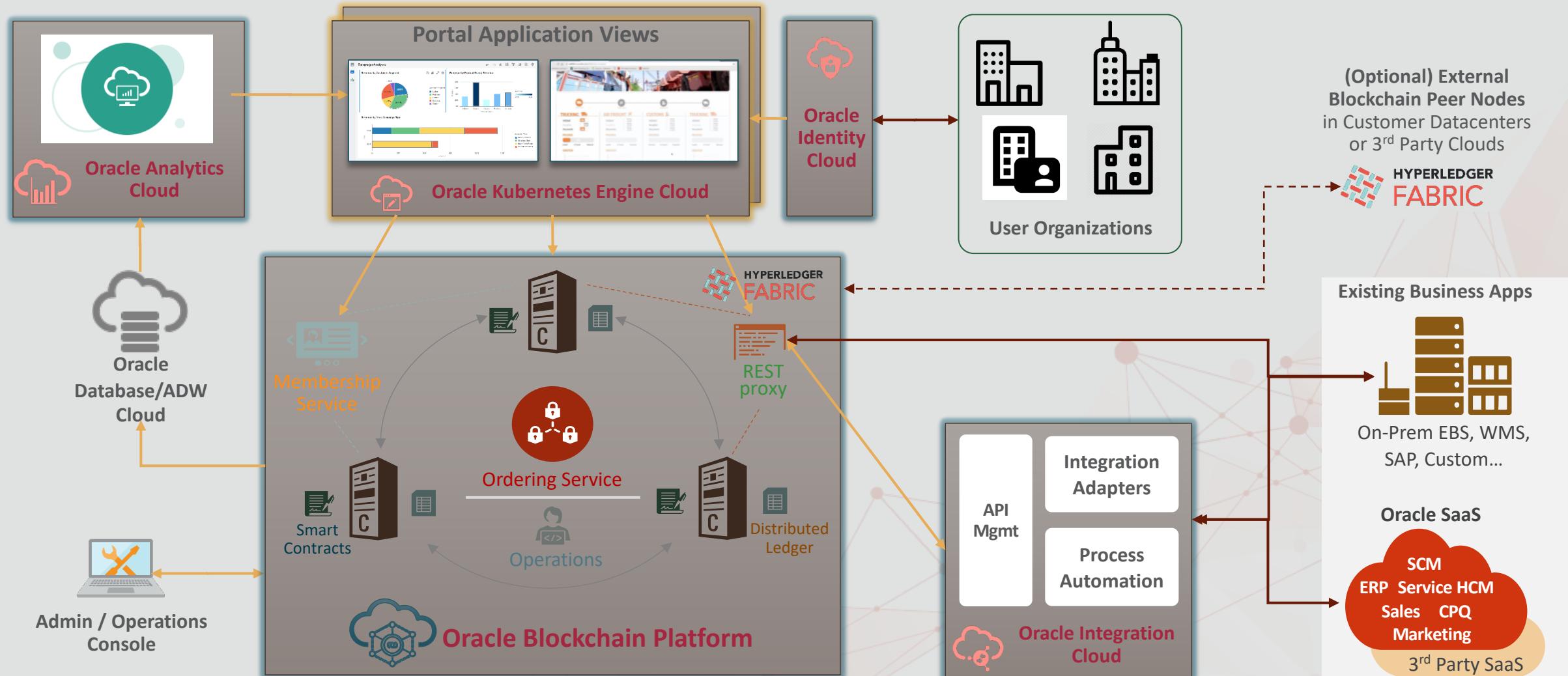
- Identity Management
(for member enrollment)
- Fault Tolerant Event Hub
(for ordering event mgmt.)
- Container Lifecycle Management
(for system and chaincode containers)
- Configuration & Monitoring Tools
- Autonomous Recovery
(when components fail)
- Managed Patching & Upgrades
- Elastic scaling on demand
- Multi-datacenter DR
(with backup of ledger and config info)



OBP High-Level Service Architecture in OCI



Sample Solution Architecture



Provisioning 비교 : Native Hyperledger vs OBP

직접 Hyperledger Fabric 설치

OS 설치

cURL install

Docker / Docker Compose 설치

Go, Node.JS설치

Python설치

Curl로 Hyperledger 설치

Crypto-config.yaml을 참조해서 Key file 생성

```
cryptogen generate --config=./crypto-config.yaml
```

Genesis block 생성

Crypto-config.yaml을 참조해서 Key file 생성

```
configtxgen -profile OneOrgOrdererGenesis -  
outputBlock ./config/genesis.block
```

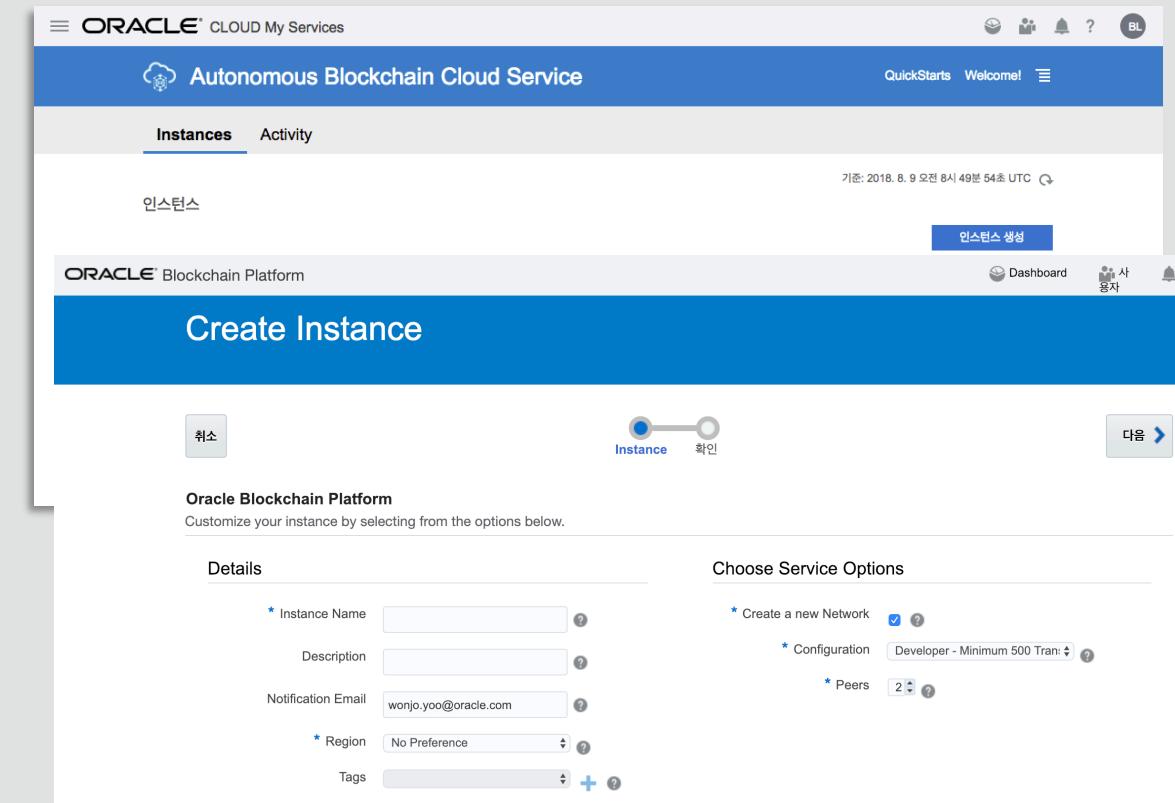
Create a Channel Configuration Transaction

```
./bin/configtxgen -profile TwoOrgsChannel -  
outputAnchorPeersUpdate ./channel-  
artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -  
asOrg Org1MSP
```

Start the network

```
docker-compose -f docker-compose-cli.yaml up -d
```

Oracle Blockchain Platform



Org 추가 및 Join 비교 : Native Hyperledger vs OBP

Hyperledger Fabric에서 구성

docker-compose-neworg.yaml 생성

Generate the New Org Crypto Material

```
..../bin/cryptogen generate --config=./org3-crypto.yaml  
cd .. && cp -r crypto-config/ordererOrganizations org3-artifacts/crypto-  
config/
```

Prepare the CLI Environment

Fetch the Configuration

```
peer channel fetch config config_block.pb -o orderer.example.com:7050 -  
c $CHANNEL_NAME --tls --cafile $ORDERER_CA
```

Convert the Configuration to JSON and Trim It Down

```
configtxlator proto_decode --input config_block.pb --type common.Block  
| jq .data.data[0].payload.data.config > config.json
```

Add the Org3 Crypto Material

```
jq -s '.[0]* {"channel_group":{"groups":{"Application":{"groups":  
{"Org3MSP":.[1]}}}}' config.json ./channel-artifacts/org3.json >  
modified_config.json  
configtxlator proto_encode --input config.json --type common.Config --  
output config.pb
```

5 more steps ...

Sign and Submit the Config Update

```
peer channel signconfigtx -f org3_update_in_envelope.pb  
peer channel update -f org3_update_in_envelope.pb -c  
$CHANNEL_NAME -o orderer.example.com:7050 --tls --cafile  
$ORDERER_CA
```

Configuring Leader Election

Join Org3 to the Channel

```
docker-compose -f docker-compose-org3.yaml up -d  
peer channel fetch 0 mychannel.block -o orderer.example.com:7050 -c  
$CHANNEL_NAME --tls --cafile $ORDERER_CA  
peer channel join -b mychannel.block
```

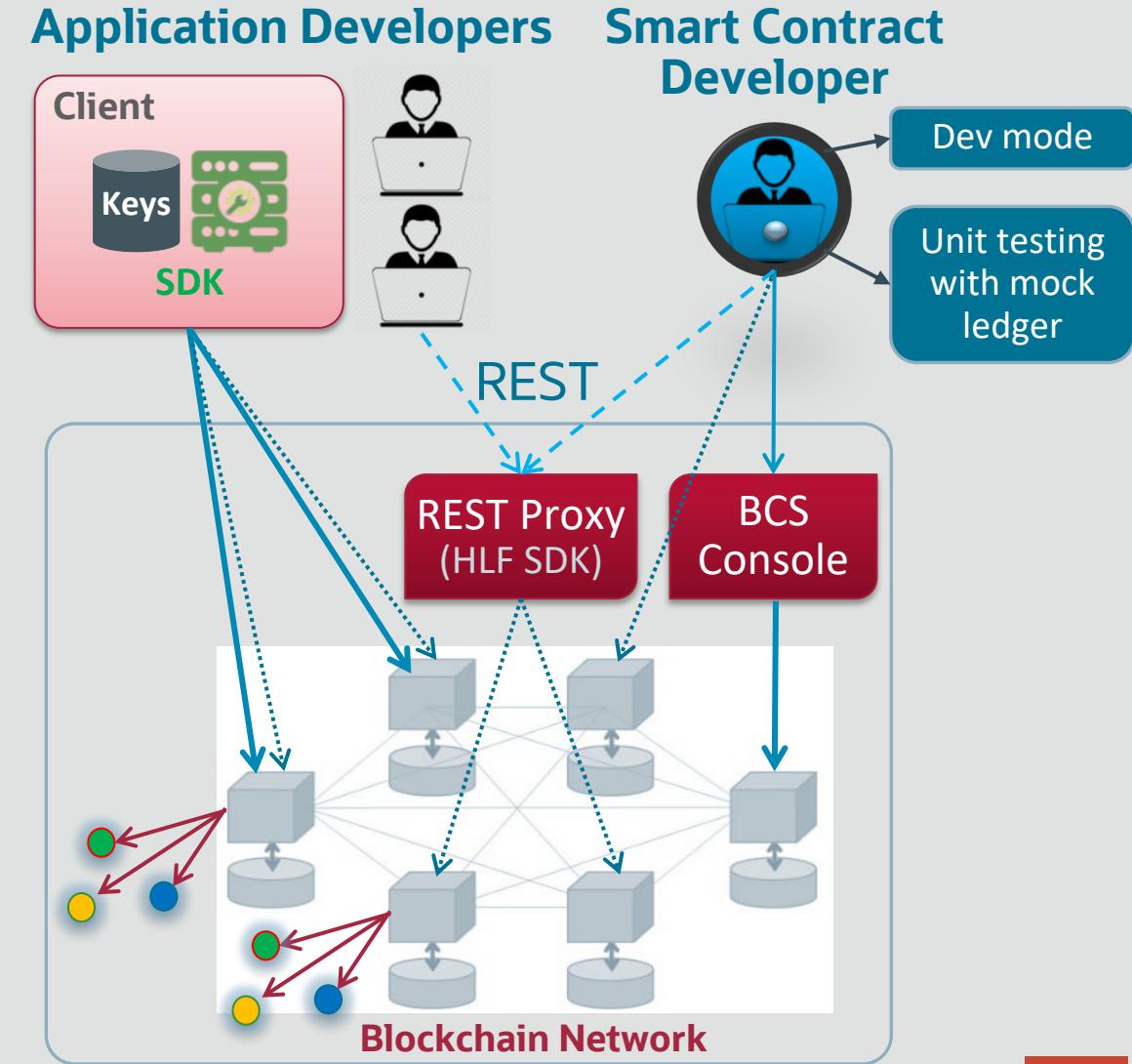
Oracle Blockchain Platform

The screenshot shows the Oracle Blockchain Platform's web interface. At the top, there are tabs for Dashboard, Network, Nodes (which is selected), Channels, Chaincodes, and Developer Tools. Below the tabs, a summary table is shown with columns for Filter, Node ID, IP, Port, Status, and Type. A red box highlights the 'Nodes' section, which lists four nodes: samdealer-peer0-1 (pink), samdealer-peer0-2 (light blue), detroitauto-peer0-1 (dark blue), and detroitauto-peer0-2 (green). Below the table, a diagram illustrates the network topology with nodes connected by lines representing channels. One channel is explicitly labeled 'samchannel' (red) and another is labeled 'default' (yellow). A modal window titled 'Join New Channels' is open at the bottom, prompting the user to 'Specify the name of the channel that you want to join.' The input field contains 'samchannel' and a 'Join' button is visible.

- 네트워크, 파트너 및 트랜잭션 관리를 위한 직관적인 관리 콘솔

Smart Contract, Application 개발

- **Smart Contract** 개발자는 블록 체인에서 비즈니스 로직을 만들고 테스트
 - 콘솔 또는 SDK를 통해 배포
 - 웹 서비스 / REST API 또는 SDK를 통한 테스트
- **Application** 개발자는 배포 된 Smart Contract를 통해 트랜잭션을 실행하는 최종 사용자 응용 프로그램을 개발
 - Client SDK (Java, Go, node.js)
 - Web Services/REST APIs
- **개발자 생산성 향상**
 - Rich queries on history DB
 - Developer Cloud를 CI / CD 용 DevOps 도구로 이용



Chaincode 배포 비교 : Native Hyperledger vs OBP

Hyperledger Fabric에서 배포

Chaincode 목록 확인

```
peer chaincode list --installed
```

```
peer chaincode list --instantiated -C org1mspc
```

Chaincode install

```
peer chaincode install -n mycc -v 1.0 -l "golang" -  
p ${CC_SRC_PATH}
```

Chaincode instantiate

```
peer chaincode instantiate -  
o ${orderer_addr}:${orderer_port} --tls --cafile  
orderer.pem -C mychannel -n mycc -l golang -v  
1.0 -c '{"Args":["init","a","100","b","200"]}' -P  
<policy_string> --logging-level debug
```

Oracle Blockchain Platform

Deploy Chaincode

Select How to Deploy



Quick Deploy

one step deployment of a new chaincode with default options. Chaincode is installed, instantiated and enabled in REST proxy.



Advanced

Step-by-step deployment of a new chaincode for full flexibility. Chaincode is installed, instantiated and enabled in REST proxy.

Deploy Chaincode (Quick)

Chaincode is installed on all peers in this instance and instantiated on specified channel(s). Default endorsement policy is used.

Chaincode Name *

Version *

Initial parameters for Chaincode Instantiation

Channel *

REST Proxy *

Chaincode Source

Deploy Chaincode (Advanced)

Step 1 of 3: Install

Chaincode is installed on target peers.

Chaincode Name *

Version *

Target Peers *

Chaincode Source

Submit

Cancel

Next

개발 편의성 - SQL기반 Query 사용

Couch DB Query Syntax : OBCS에서 호환됨

```
func (t *CarChaincode) queryUserByNamePhone(stub shim.ChaincodeStubInterface, args []string) peer.Response {  
    queryString := fmt.Sprintf("{\"selector\":{\"UserName\":\"%s\",\"PhoneNo\":\"%s\"}}", name, phone)
```

Berkeley DB Query Syntax : SQL 형태로 되어 있어 사용이 편리함

```
queryString := fmt.Sprintf("SELECT valueJson FROM <STATE> WHERE json_extract(valueJson, '$.docType', '$.UserName', '$.PhoneNo')
```

```
func (t *HanaChaincode) queryUserByIdLike(stub shim.ChaincodeStubInterface, args []string) peer.Response {  
    queryString := fmt.Sprintf("SELECT valueJson FROM <STATE> WHERE key like '%s%%'", id)
```



계정 관리 시스템 연계

- Oracle 클라우드의 계정관리 시스템과 Hyperledger의 CA와 계정 연계
- Third Party 계정 Provider와의 연동 - SAML지원, Microsoft Active Directory 연동, SSO
- 사용자별 ACL 관리

The image displays two screenshots of the Oracle Identity Cloud Service interface.

Left Screenshot: Shows the application management screen for 'OABCINST_SeoulAuto'. It lists several roles and their associated users:

- ORDERER (BCS Orderer service 관리자 풀): 지정된 사용자 1명
- ADMINISTRATOR (CA administrator 관리자 풀): 지정된 사용자 1명
- CLIENT (CA client 관리자 풀): 지정된 사용자 1명
- BCS USER (BCS user): 지정된 사용자 1명
- RESTPROXYO_ADMIN (RESTPROXYO administrator 관리자 풀): 지정된 사용자 1명
- BCS ADMINISTRATOR (BCS administrator): 지정된 사용자 1명

Right Screenshot: Shows the process of adding a SAML 2.0 ID provider. The steps are:

- 세부정보 (Detailed Information)
- 구성 (Configuration)
- 매핑 (Mapping)
- 익스포트 (Export)
- 테스트 (Test)
- 활성화 (Activation)

The current step is '구성' (Configuration). A form is filled out with the following details:

- * 이름 (Name): [Empty input field]
- 설명 (Description): [Empty input field]
- 아이콘 (Icon): [Empty input field with a file upload icon]
- 업로드 (Upload): [Upload button]

Rich History DB for Analytics Integration

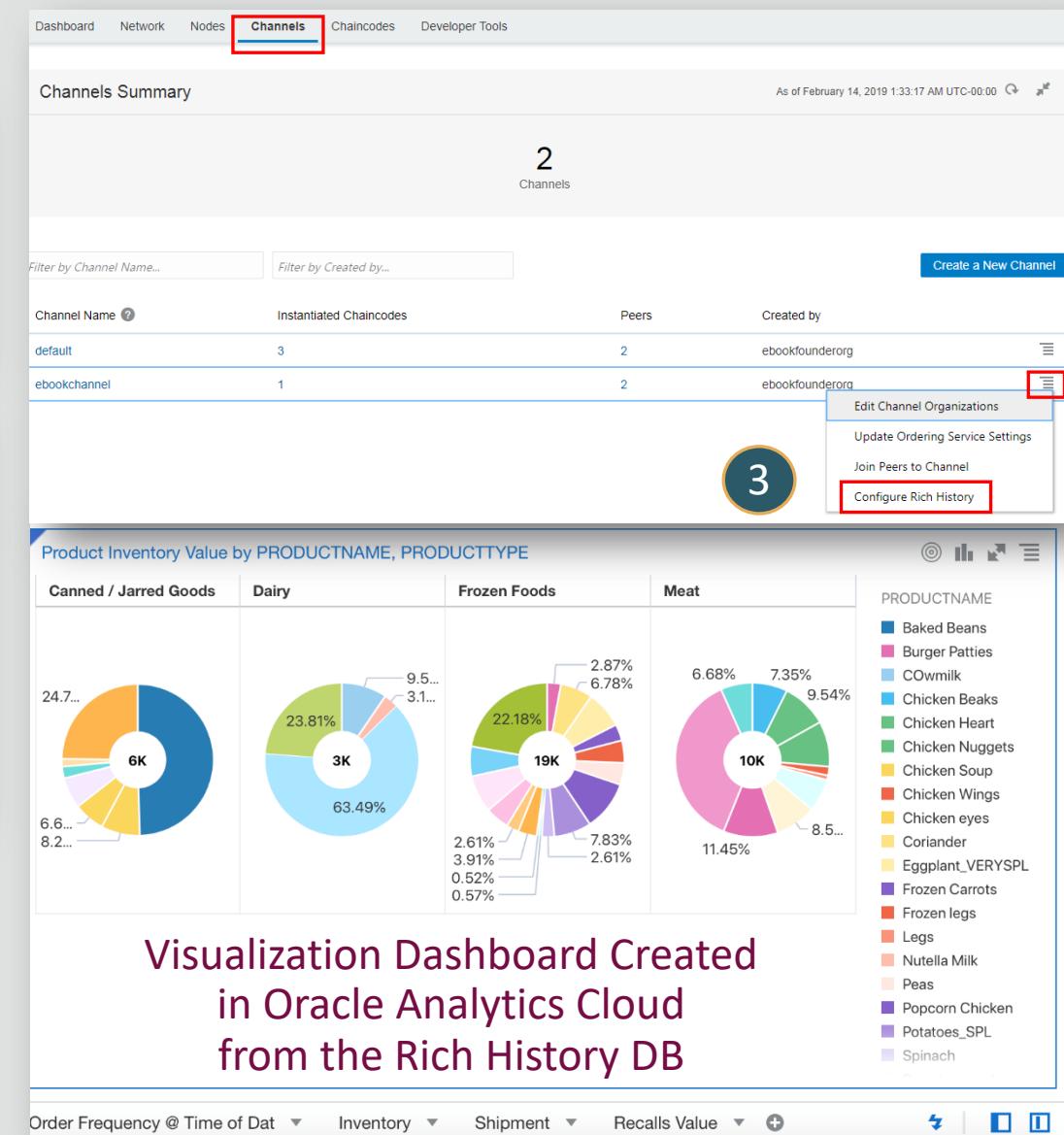
Parallel with regular history DB updates, we asynchronously update Oracle ADW/DBaaS for every transaction commit

DB maintains rich data model using Oracle JSON support (can be unpacked in OAC project)

Accessible for Analytics / BI / DWH reporting, interactive visualization dashboards, etc.

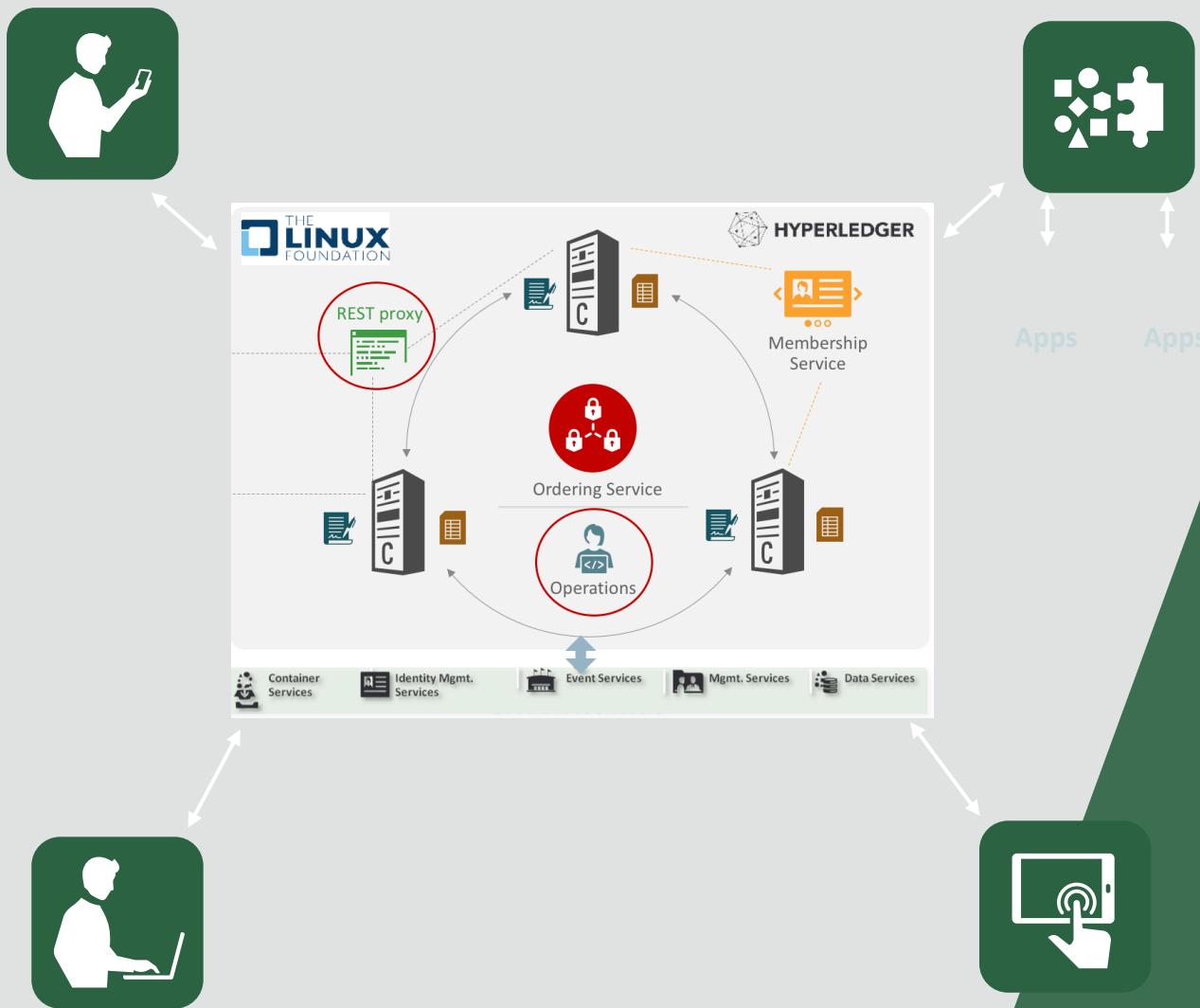
Can be used for transaction confirmations and high volume read-only access when async delay is not critical

The screenshot shows the Oracle Analytics Cloud interface. A user named 'cloud.admin' is logged in. A prominent button labeled 'Configure Rich History' is highlighted with a red circle labeled '1'. Below it, another window titled 'Configure Rich History' is open, also highlighted with a red circle labeled '2'. This window contains fields for 'User Name' (admin), 'Password' (redacted), 'Connection String' (minoloadw_high), and 'Wallet package file(Optional)' with a 'Upload wallet file' button.



Visualization Dashboard Created
in Oracle Analytics Cloud
from the Rich History DB

Easily Integrate and Develop Applications



- Oracle SaaS SCM 및 ERP 애플리케이션 확장을 위한 사용 준비된 SDKs
- Oracle 또는 타사 애플리케이션에서 블록체인 트랜잭션을 트리거하는 간편한 REST API
- Oracle PaaS 서비스에 대한 블록체인 애플리케이션을 신속하게 개발

Oracle Apps Leveraging Blockchain Platform

API-Based Integrations



40,000+ Global Organizations



319 Banks in more than 115 countries



Oracle's PaaS/SaaS curated Fintech AP



Brand Compliance in Grocery Supply Chain



Clinical trial data collection & reporting



SaaS on OBP



Intelligent Track and Trace

Lot Lineage and Provenance

Intelligent Cold Chain

Warranty and Usage Tracking

OABC Smart Contracts, Distributed Ledger

Shipment Notifications, Bill of Lading, Manufacturing work orders

Purchase order, Sales order

Service records, Warranty information

Authentication, Immutability, Trust, Consensus

Cargo conditions, Predictive Insights

SOURCES OF INPUT



Customer Driven



*coming soon



Oracle Intelligent Track and Trace

→ C ⌂ oracle.com/webfolder/s/quicktours/blc/gqt-blc-itt-overview/index.html

Oracle Blockchain Applications Request a Live Demo

Oracle Intelligent Track and Trace

Alpha Electronics is transporting from the West Coast USA across the Atlantic.

13 of 20

The screenshot displays the Oracle Intelligent Track and Trace application interface. On the left, a vertical timeline shows a sequence of shipping events for a purchase order (BPO-8467) on May 14, 2019. The events are color-coded by location: purple for the US West Coast, green for the Atlantic Ocean, blue for Europe, and red for the South American coast. The events include:

- Shipping (30972) from Vision Computers to Delta Audio
- Purchase Order (AOPO-1673) from Alpha Electronics to Zeta Carrier
- Bill Of Lading (BSOL-1426) from Zeta Carrier to Beta Batteries
- Production Report (3386) from Alpha Electronics to Delta Audio
- IoT Fleet Monitoring Event (BOT-0597) from Zeta Carrier to Alpha Electronics
- Shipping (75444) from Alpha Electronics to Vision Computers
- Material Receiving Receipt (11129) from Vision Computers to Speedy Carrier
- Bill Of Lading (ASOL-8967) from Speedy Carrier to Alpha Electronics
- QA Inspection (1234) from Vision Computers to Vision Computers
- Material Receiving Receipt (59317) from Vision Computers to BigBox Retailer
- Purchase Order (TPO-79) from BigBox Retailer to Vision Computers
- Sales Order (TSO-18) from Vision Computers to Vision Computers

The central part of the interface features a world map showing the shipping route. A pink dashed line traces the path from the United States (West Coast) through the Atlantic Ocean to South America. Blue location markers are placed along this line, corresponding to the events listed in the timeline. The right side of the interface contains a detailed sidebar with "Details" about the transaction, including:

- ORDER AUDIO COMPONENTS Ship Product
- DOCUMENT TYPE Shipping
- DOCUMENT NUMBER 75444
- LEDGER KEY Shipping:75444
- DOCUMENT RECEIVED 08:21:26 am
- FROM TRADING PARTNER Alpha Electronics
- TO TRADING PARTNER Vision Computers
- DOCUMENT DATA
- TRANSACTION VALUE 3000 USD
- REFERENCED ITEMS
- DOCUMENT CROSS REFERENCE

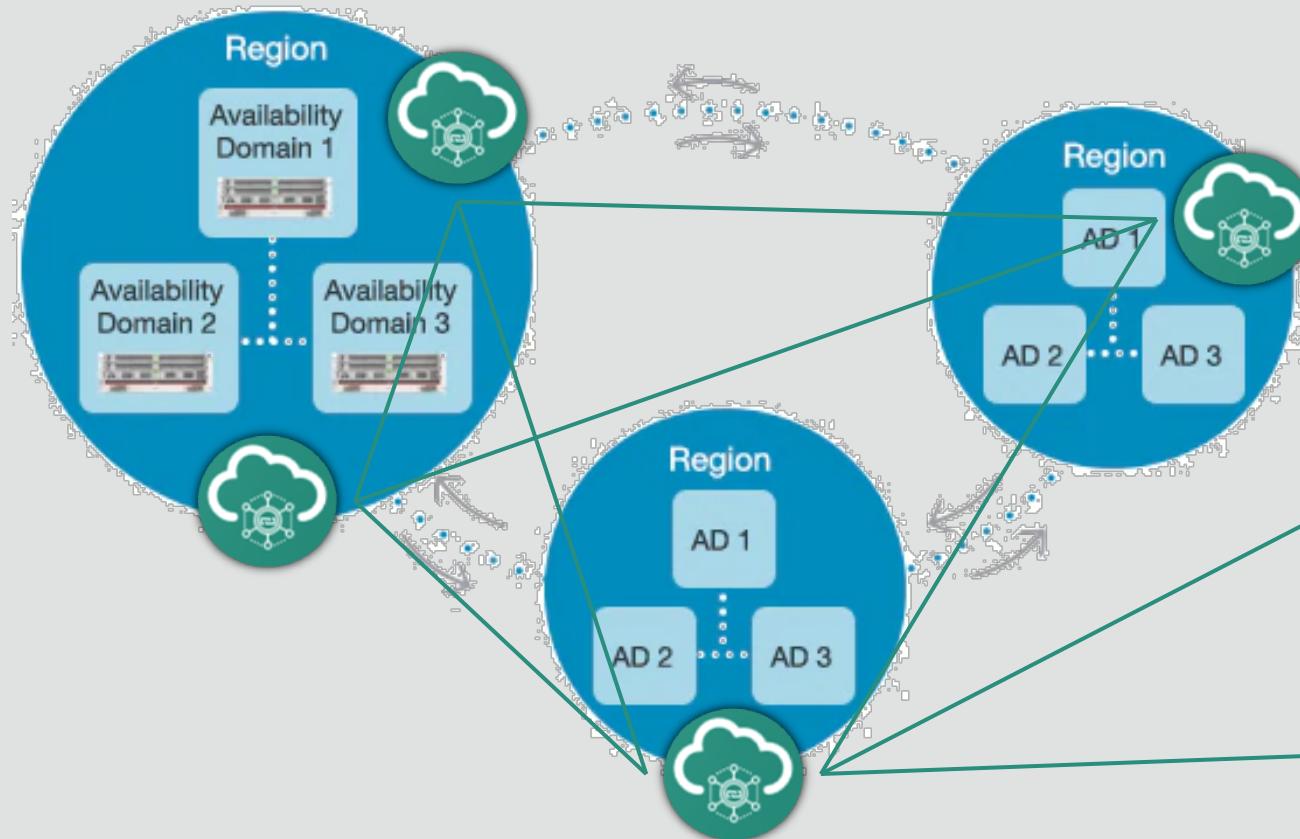
QuickStart Templates



	Developer	Enterprise X1	Enterprise X4
BCS Control VMs	1	2	2
Chaincode VMs	1	1	1
VM Shapes	2.1 (1 OCPU)	2.1 (1 OCPU)	2.4 (4 OCPU)
Kafka	1 node in control VM	3 node cluster in 3 VMs	3 node cluster in 3 VMs
Max Peer Nodes	4	14	14
Orderer Nodes	1	2	2
Storage	1TB Block + 1TB OSS	2TB Block + 1 TB OSS	2TB Block + 1 TB OSS
Minimum Consumption	1 unit (500 txn/hr)	2 units (1000 txn/hr)	8 units (4000 txn/hr)
Minimum Monthly Flex	\$372	\$744	\$2,880

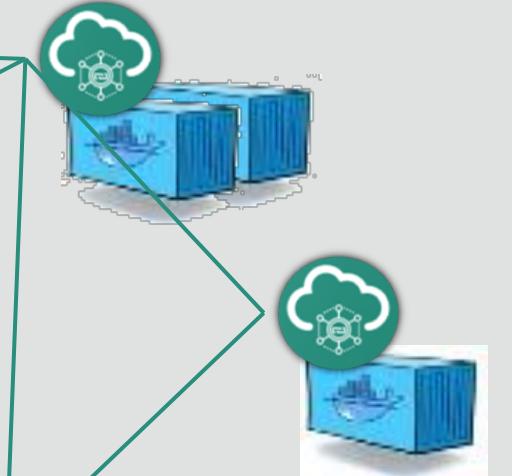
Flexible, Global Blockchain Network Topologies

Oracle Cloud Datacenters



Customer Datacenters

Oracle Blockchain Platform
On-Premise/Private Cloud



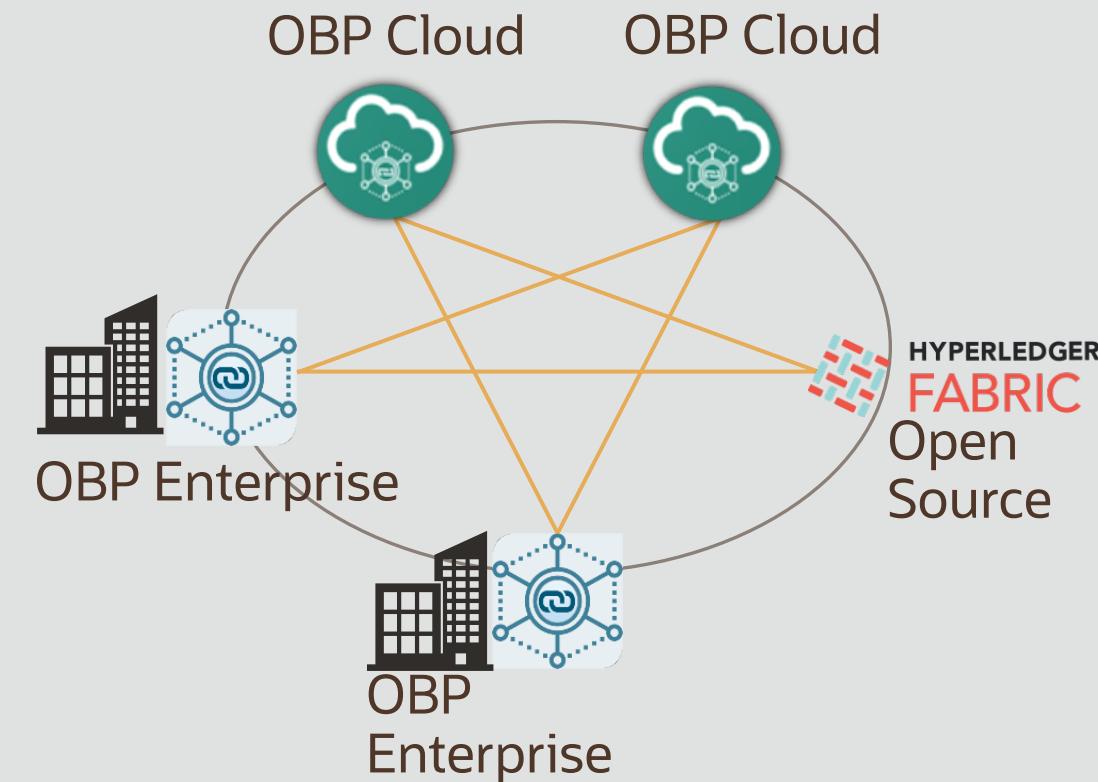
 HYPERLEDGER FABRIC
Open source deployed
on-premises or on a
3rd party cloud

*coming soon



Oracle Blockchain Platform Enterprise Edition

- Deploy Oracle Blockchain in your data center
 - Choice of virtualization platforms
- Create Blockchain network with a few clicks
 - Fully pre-assembled
- Feature parity with Blockchain Cloud
 - Same APIs & portability of applications
- Support for hybrid networks across
 - Oracle Cloud & 3rd party Blockchains



Oracle Blockchain Platform Enterprise Edition

- 1. Blockchain Platform Manager** : quick provisioning, patching of your blockchain instances
All components required to create and run a Hyperledger Fabric-based blockchain network
- 2. Oracle Blockchain Platform Console** : managing, configuring, and monitoring each instance using Web UI or an extensive set of REST APIs
- 3. Oracle Blockchain REST Proxy** : simple REST APIs to invoke transactions, query the ledger, and subscribe to events
- 4. OpenLDAP** : out-of-the-box identity management and support for using an external LDAP Directory Service

Supported Hypervisors:

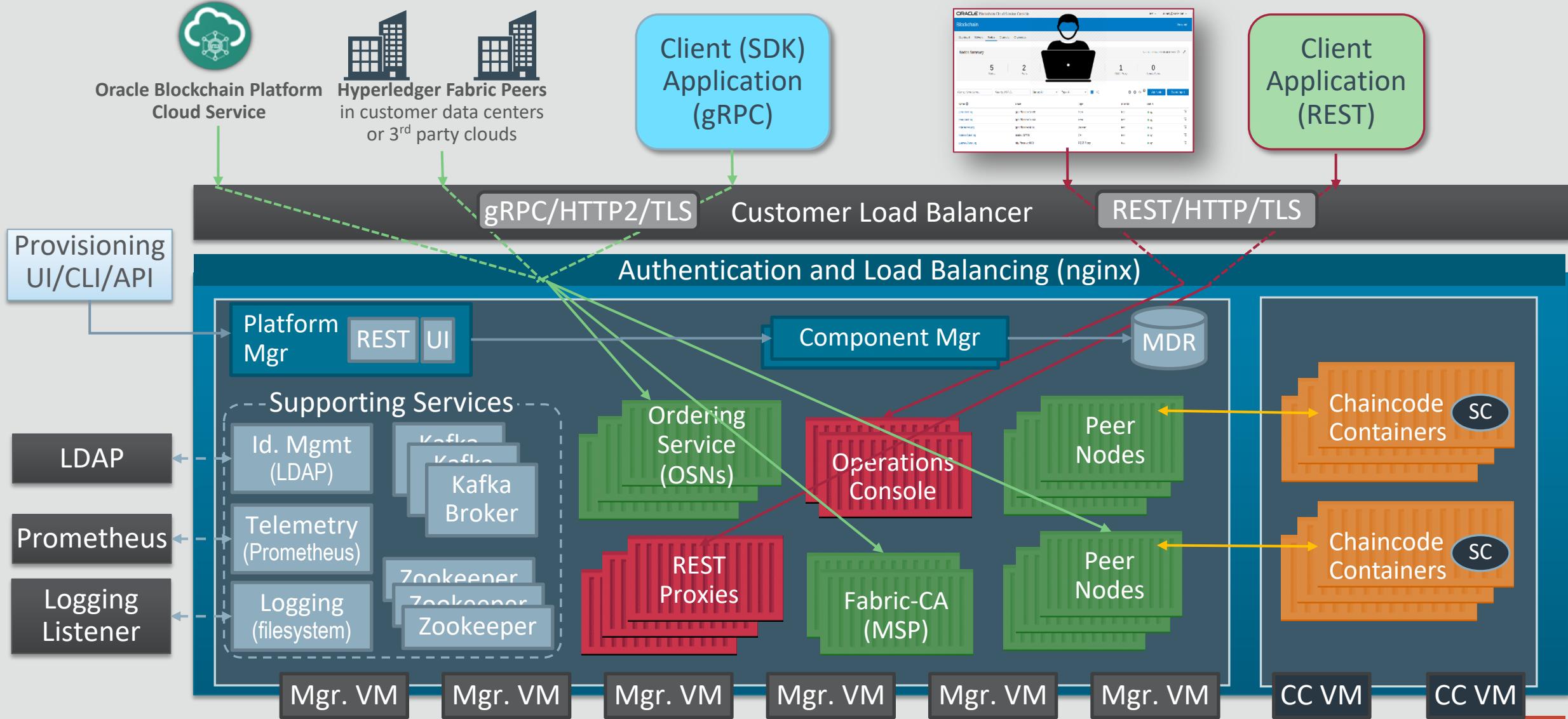
Oracle VirtualBox, versions 5.x and 6.0 or later

VMware vSphere ESXi, version 6.7 or later

Oracle Linux Virtualization Manager v4.2.8.2-1.0.8.el7 or later



OBP Enterprise Edition Architecture



Oracle Blockchain Platform - Delivering Enterprise Grade Capabilities



Development & Integration

- SQL-based rich queries
- Synchronous & async REST APIs
- Java, GO, and Node.js SDKs
- Transaction history and ledger replication in relational database
- Enhanced REST Proxy Clustering



Performance at Scale

- Parallel transaction execution
- Dynamic scale-out
- Much faster world-state DB



Supportability & Operations

- Dynamic configuration
- Monitoring dashboard
- Zero-downtime managed patching
- Autonomous recovery agents
- REST APIs for maint. and admin
- Full DR enablement



Security & Confidentiality

- Integrated identity management
- OOTB Integration with Directory Services
- Data encryption at-rest
- Certificate revocation management
- REST proxy SSO and multi-identity support
- Support for 3rd party intermediate CA



Deployment

- Choice of Virtualization Platform: VirtualBox or VMWare
- Preassembled, a few click deployment: Ready to deploy apps in mins
- High availability VMs

■ Oracle Cloud Feature
■ Enterprise Edition Feature

Thank you

Wonjo Yoo

Principal Solution Engineer
Cloud Excellence Team

