하이퍼레저 패브릭 기반의 프라이빗 블록체인 개발 과정

2020-10-19

빅픽처랩㈜

안 휘

강사 소개

- **블록체인기반 소프트웨어 시스템 설계 및 개발 전문가**

안 휘
E. hwi.ahn@bigpicturelabs.io
M. 010-9426-8501

- **경력**
  - 빅픽처랩(주), CTO, 2018.5 ~ 현재
  - KAIST, 전산학부, 박사과정, 2012.3 ~ 현재(...?)
  - CMU, MSIT-SE, 석사, 2012
  - KAIST, 전산학과, 학사, 2010
- **대표 프로젝트**
  - **잇닷 & trust-chain**: 블록체인 기반 오피니언 보드 소프트웨어 서비스
    - Hyperledger Fabric, mongoDB, Server using Go, Web/Mobile using TypeScript/JavaScript
  - **it-chain**: 오픈소스 경량 블록체인 엔진
- **Background**
  - Software Architecture & Software Engineering

전체 목차

- **하이퍼레저 패브릭 개요 (1 day)**
  - 블록체인의 본질
  - 프라이빗 블록체인의 현재
  - 하이퍼레저 패브릭을 구성하는 기술들
  - 하이퍼레저 패브릭 구동 실습
- **하이퍼레저 패브릭 개발 (2 day)**
  - 스마트컨트랙트 개요
  - 하이퍼레저 패브릭 아키텍처
  - 하이퍼레저 패브릭 스마트컨트랙트 개발 실습

블록체인의 본질

2020-10-19

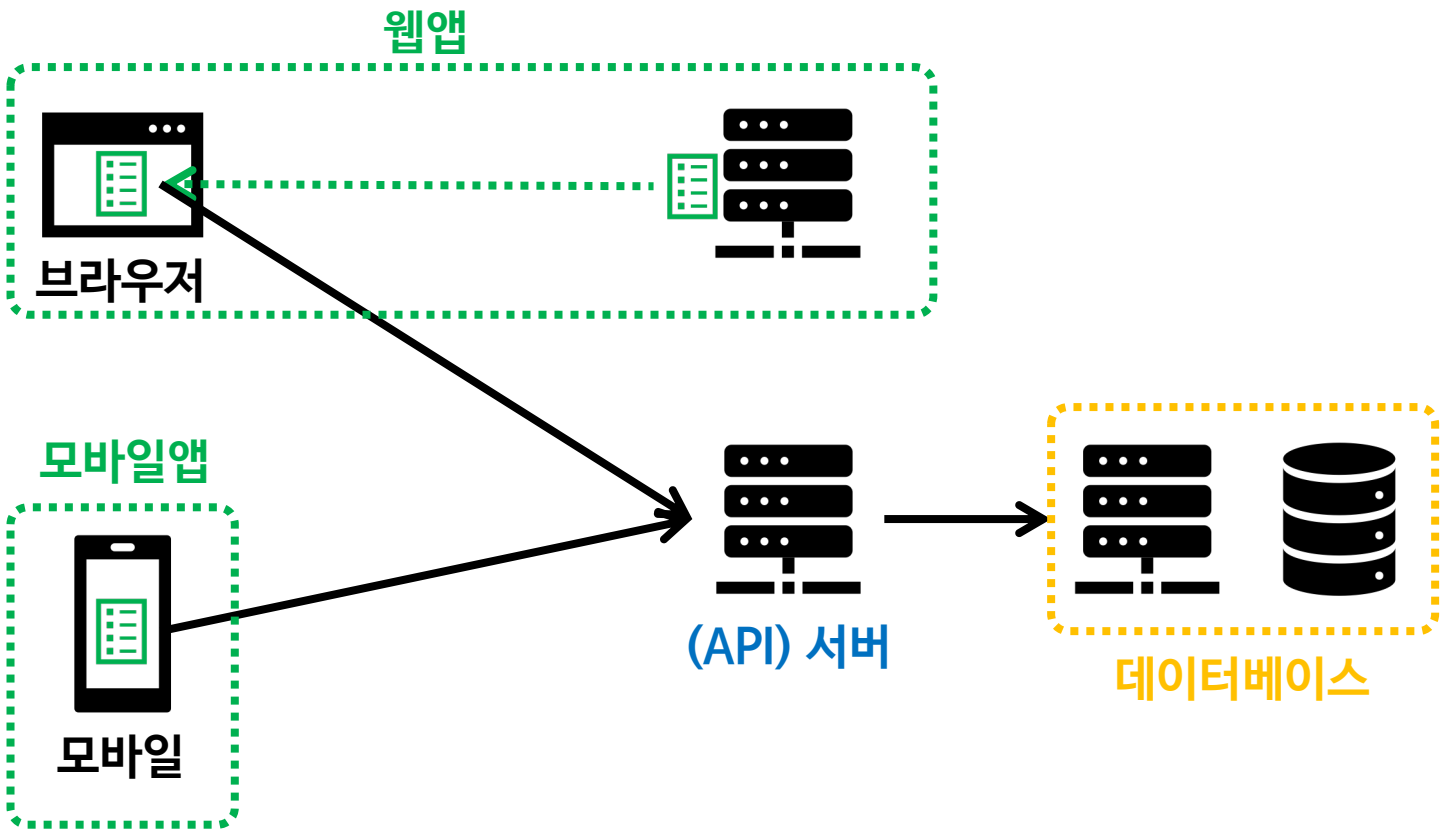
빅픽처랩(주)

안휘

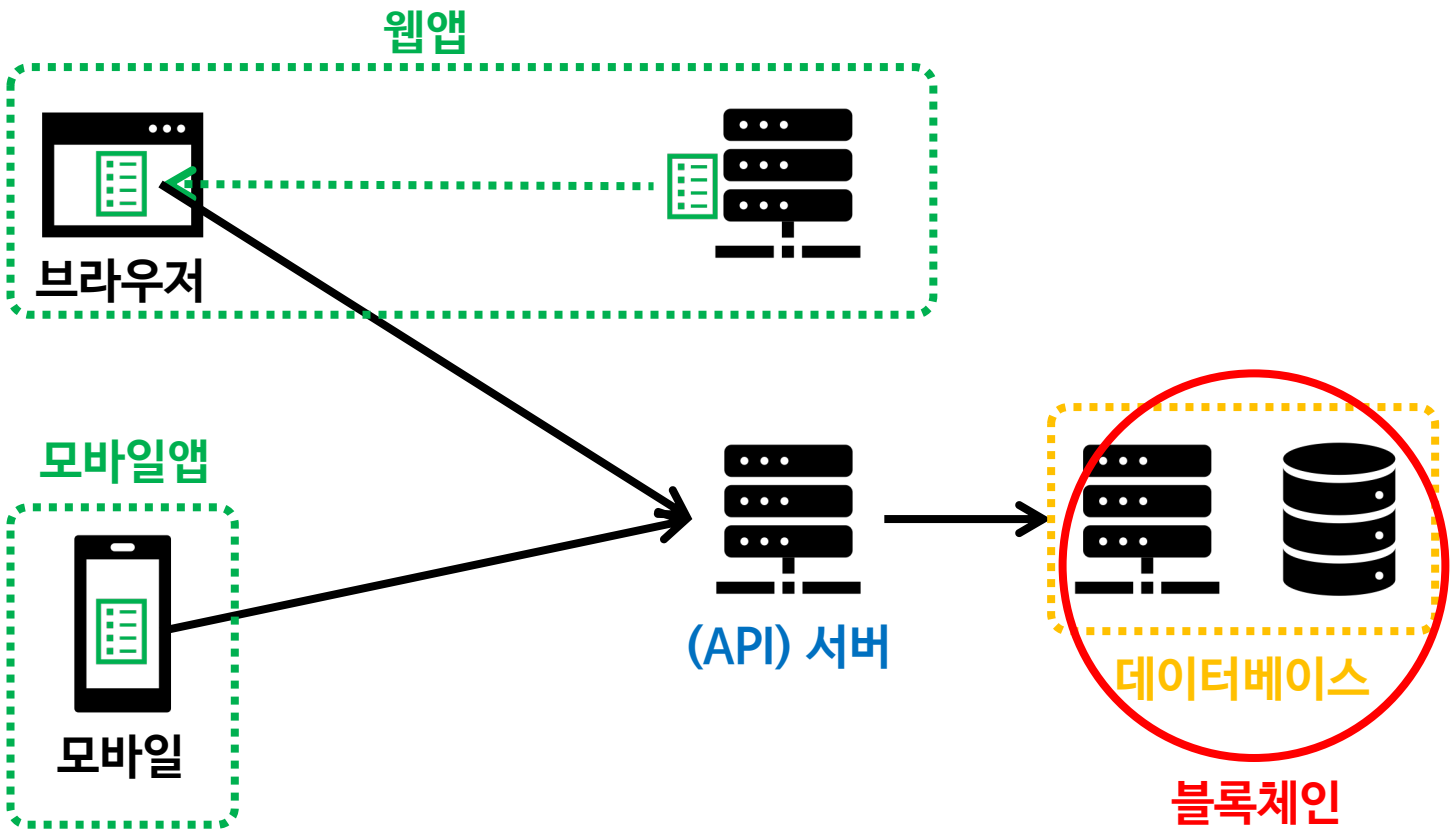
목차

1. 소프트웨어 서비스의 실제
2. 블록체인 기술
3. 블록체인 기술의 구성요소
4. 블록체인 기술의 장점
5. 블록체인 기술의 한계

1. 소프트웨어 서비스의 실제



1. 소프트웨어 서비스의 실제



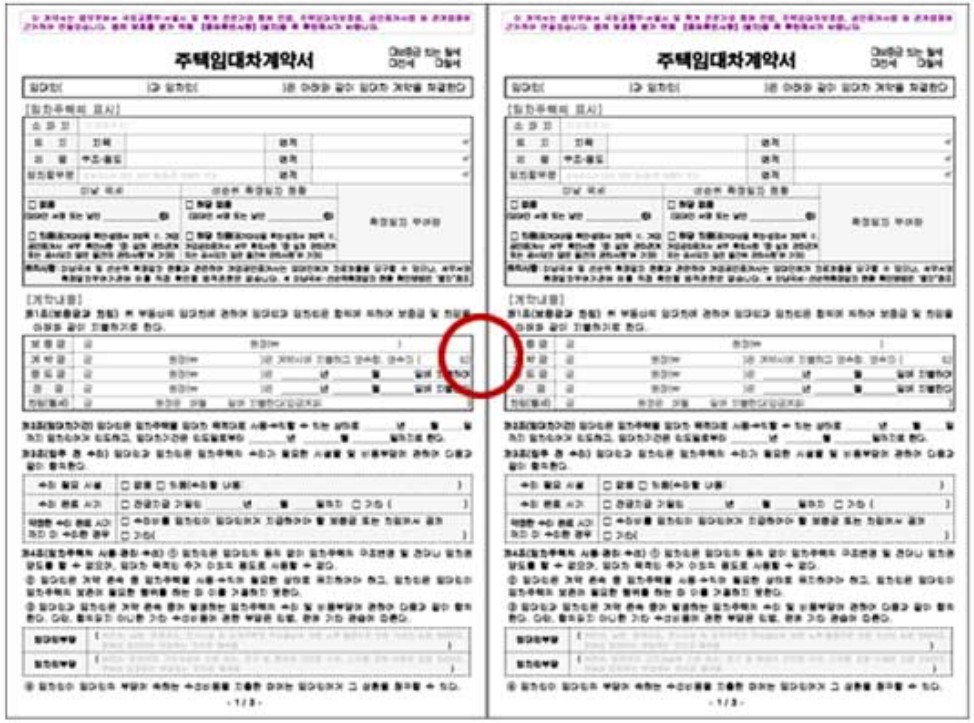
2. 블록체인 기술

인간적 신뢰가 구축되지 않은 관계자들 사이에서
중요한 데이터를
신뢰할 수 있는 방법으로 공유(저장)하는 기술

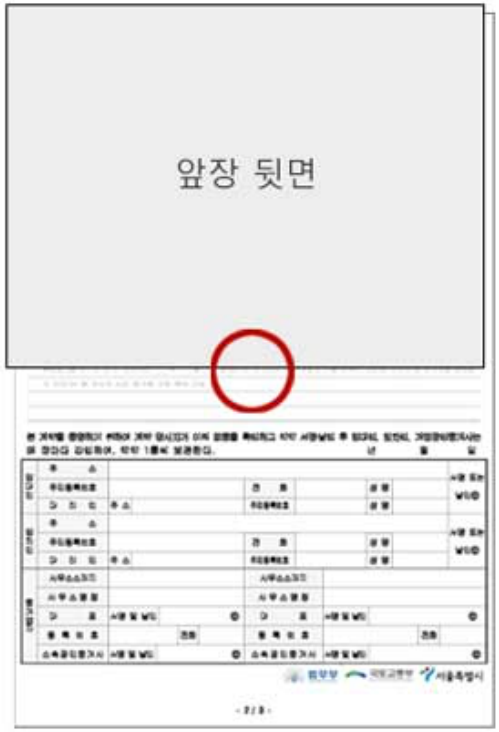
분산 데이터베이스 기술

2. 블록체인 기술

- 부동산 계약서

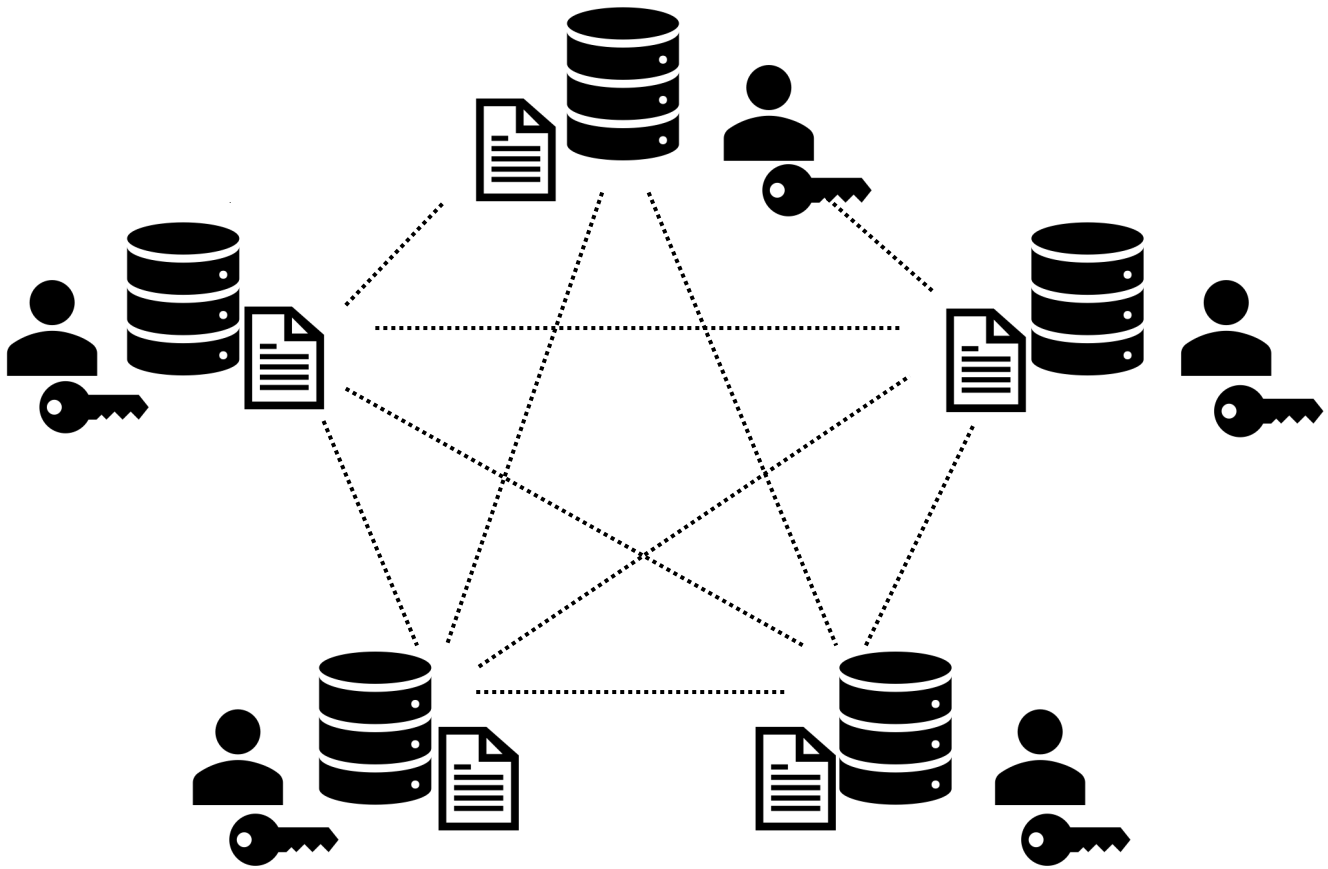


할인(割印)

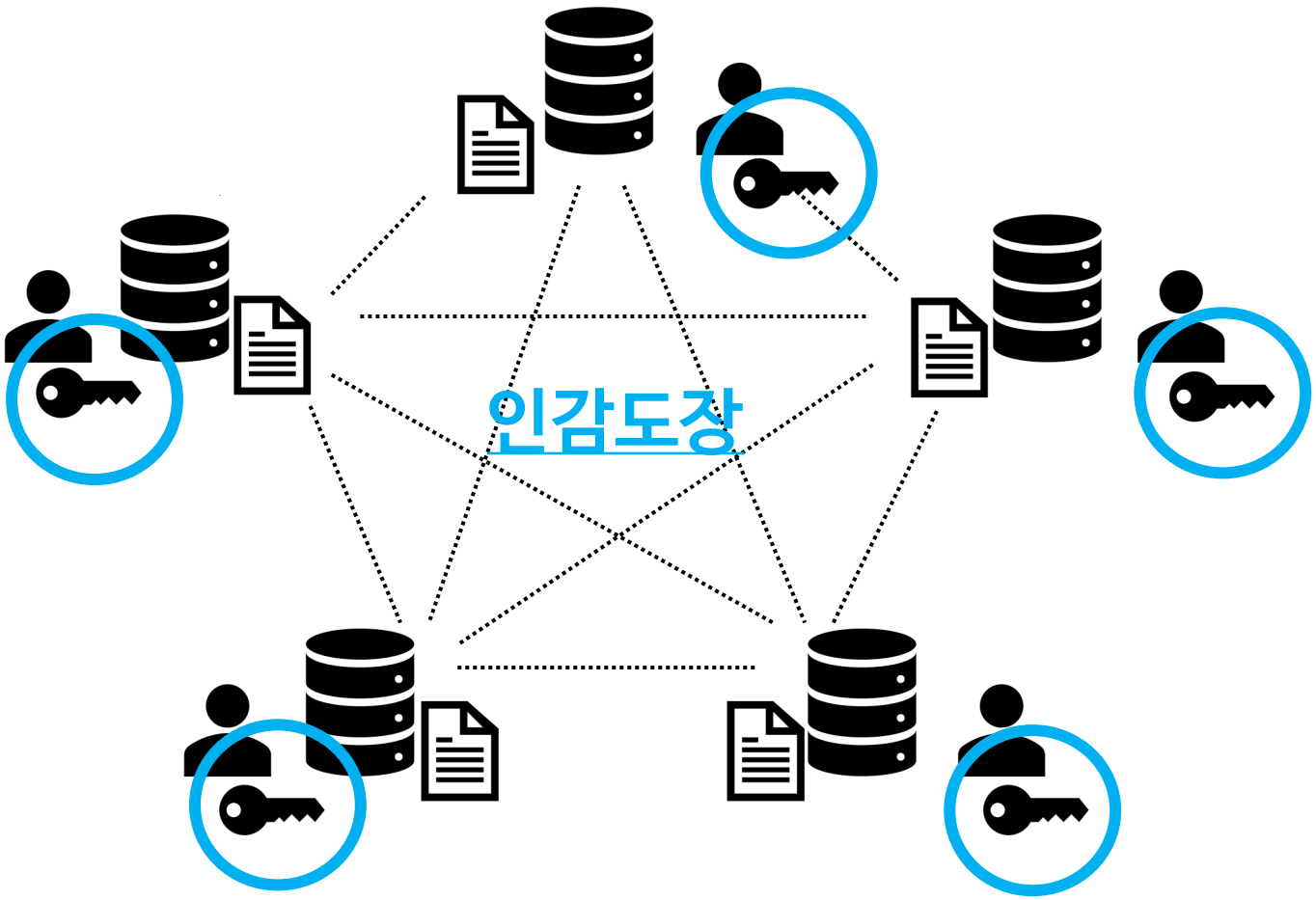


간인(間印)

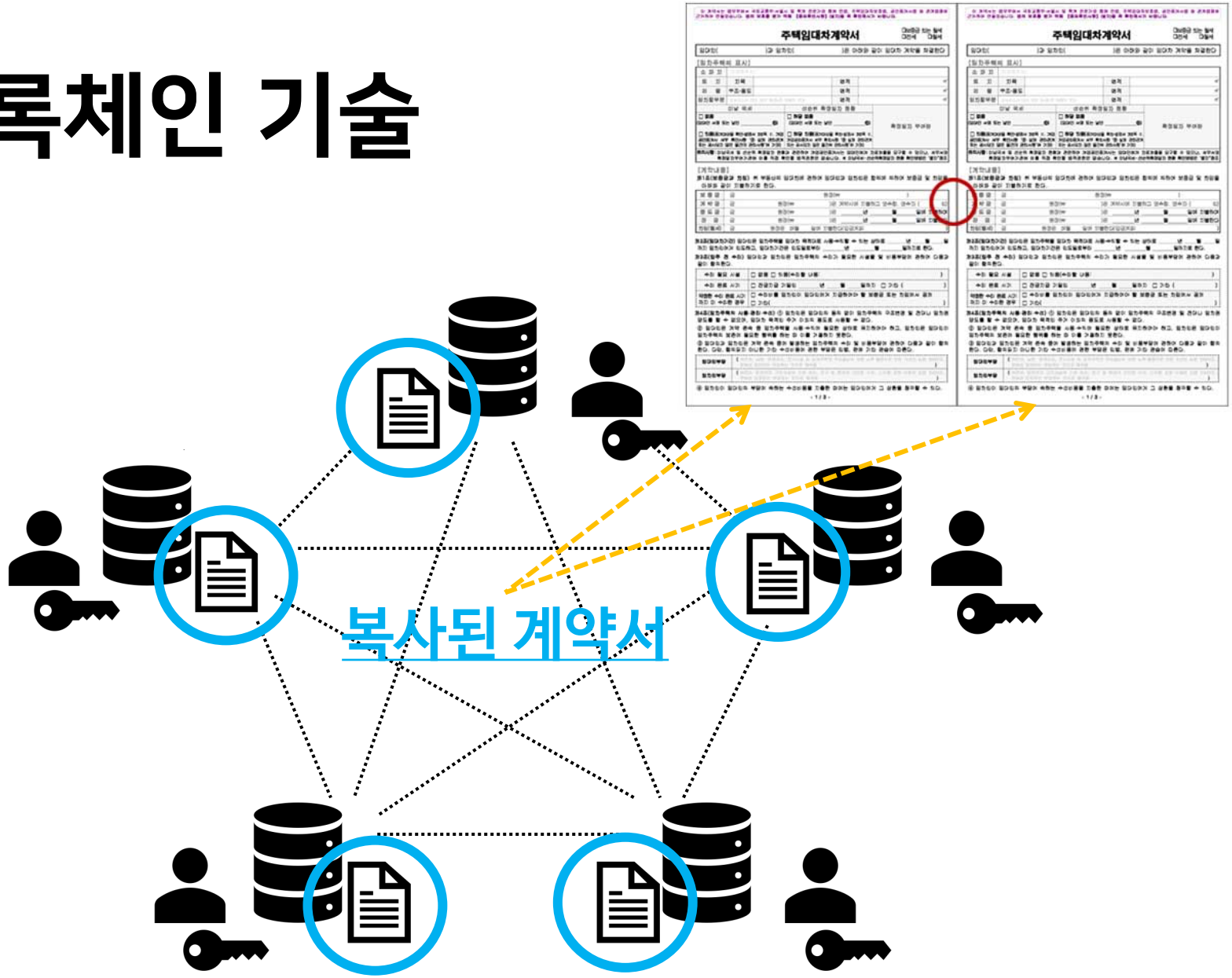
2. 블록체인 기술



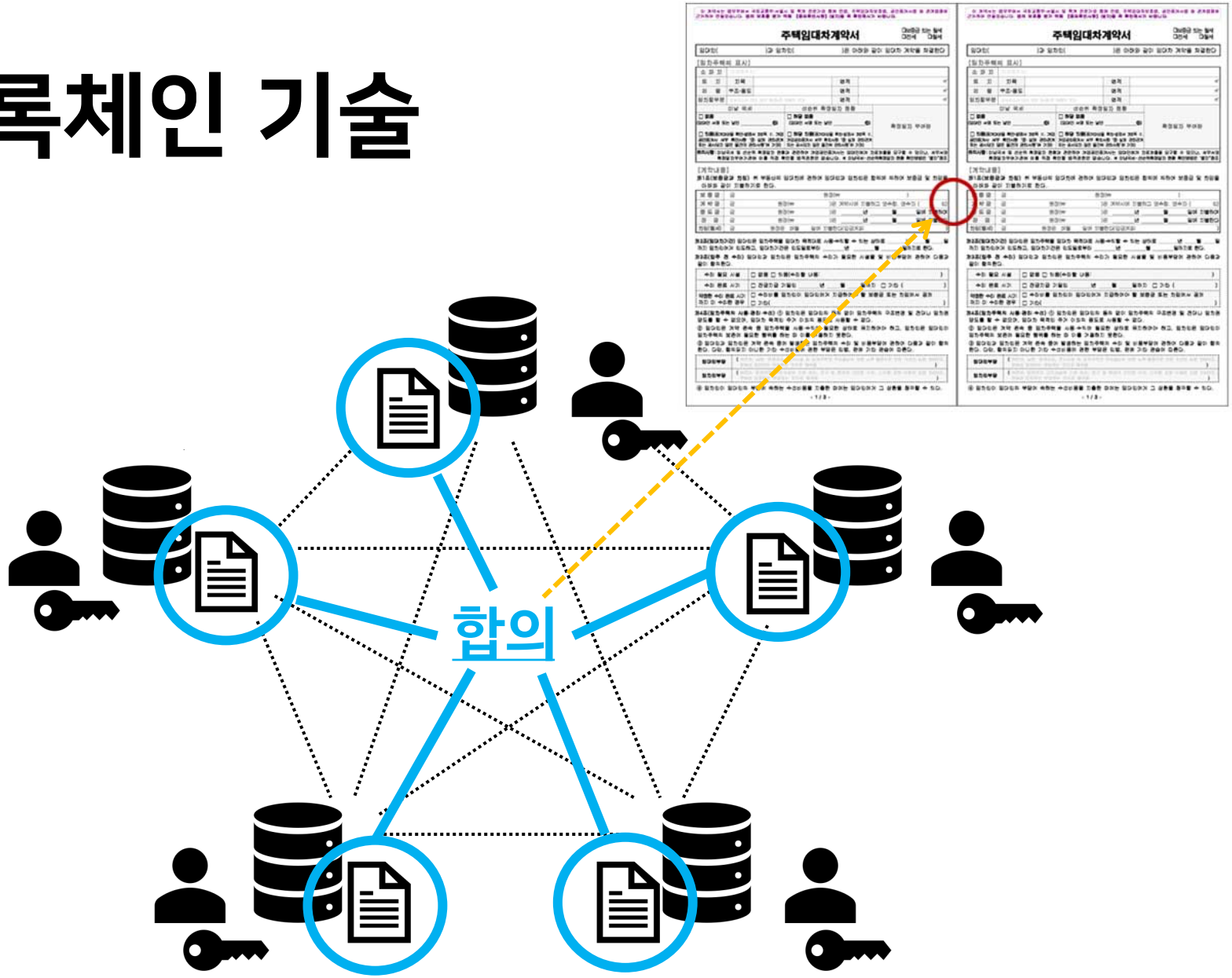
2. 블록체인 기술



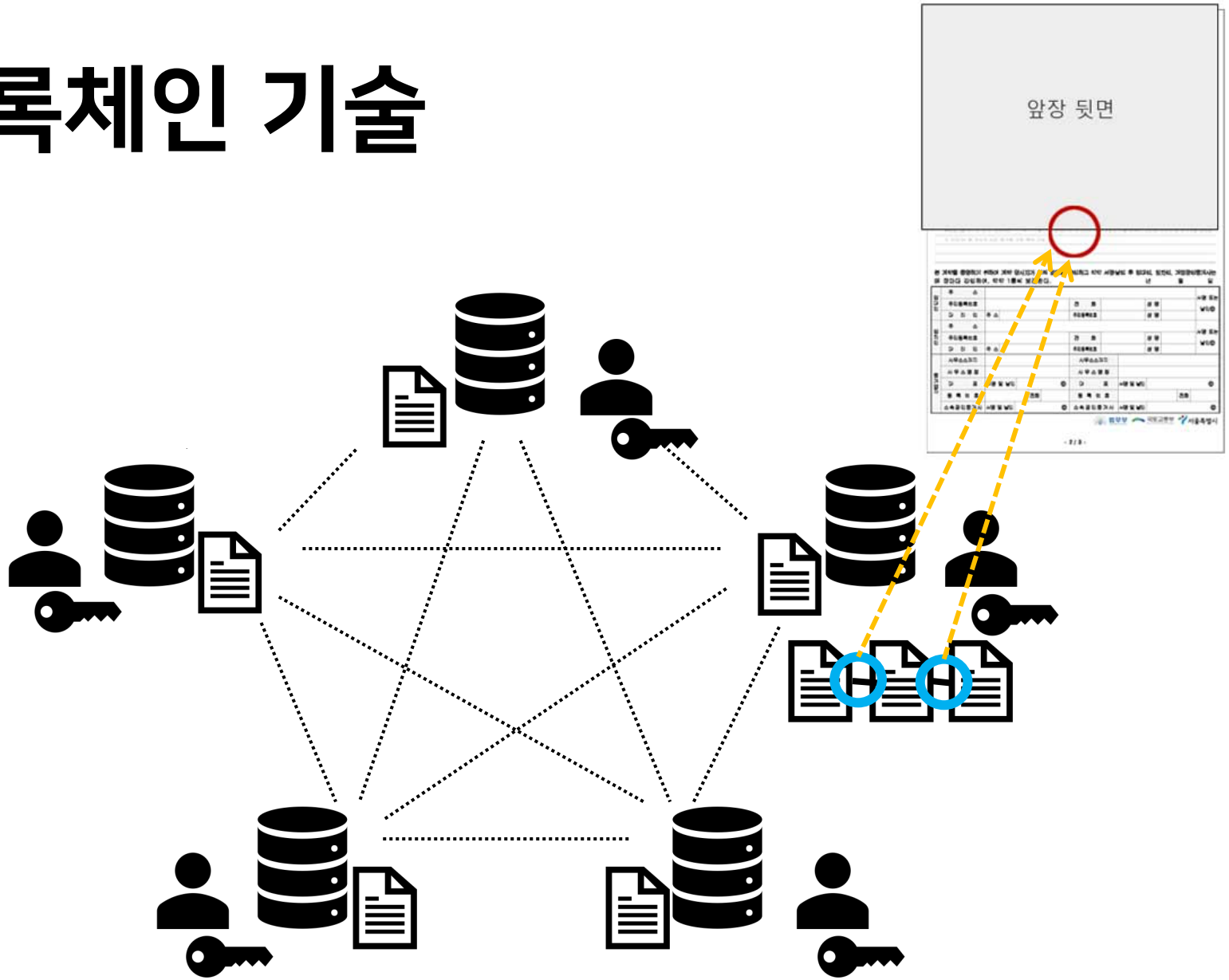
2. 블록체인 기술



2. 블록체인 기술

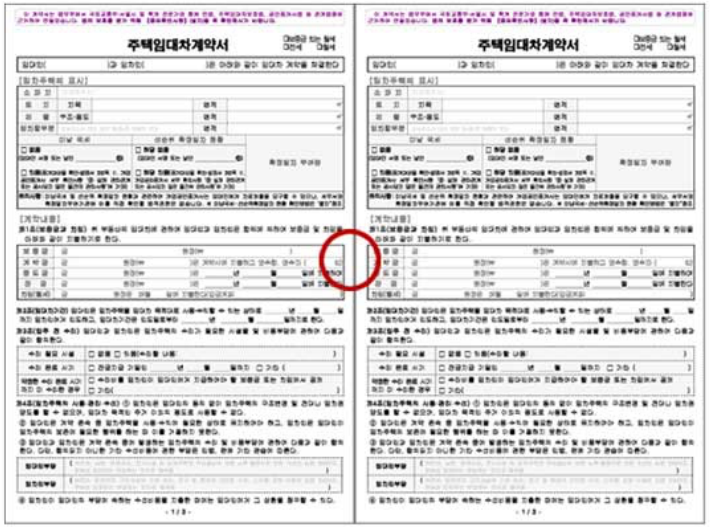


2. 블록체인 기술

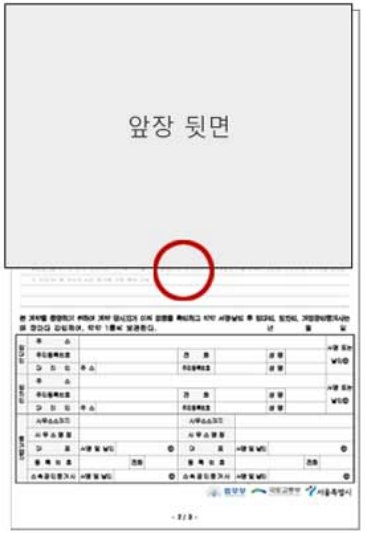


3. 블록체인 기술의 구성요소

- **부동산 계약서 구성요소**
  - **인감도장**: 내가 바로 내가 맞는 것을 증명
  - **참여자 숫자만큼 복사한 계약서**: 실제 데이터
  - **할인**: 참여자들 각각이 갖고 있는 데이터가 모두 동일함을 확인
  - **간인**: 데이터의 순서가 중간에 위변조되지 않음을 확인



할인(割印)



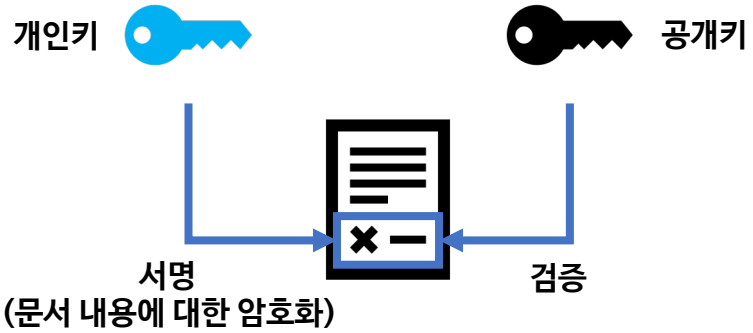
간인(間印)

3. 블록체인 기술의 구성요소

- **인감도장: 공개키/개인키 기술 (PKI)**

  - 부동산 계약서의 **인감도장**
    - 정부가 보장해줌 (인감 등록, 인감증명서 등)

  - 블록체인의 인감도장: 공개키/개인키
    - 개인키로 문서를 암호화
    - 쌍을 이루는 공개키만으로 풀 수 있음
      -〉 해당 개인키를 갖고 있는 사람이 보낸 문서가 맞는 것을 검증
    - "**공인인증서**"의 기반 기술



온라인 데모

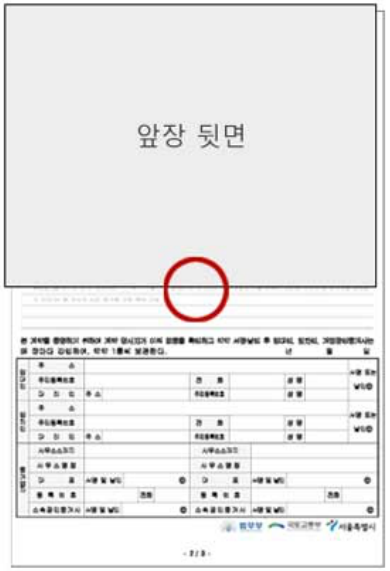
3. 블록체인 기술의 구성요소

- **간인: 해시 기술 (Hash)**
  - 해시
    - 단방향 암호화 기술
    - 특정 문서를 복구 불가능한 문자열로 암호화해줌

Hello, World! —해시!→ DFFD6021BB2BD5B0AF676290809EC3A53191DD81C7F70A4B28688A362182986F

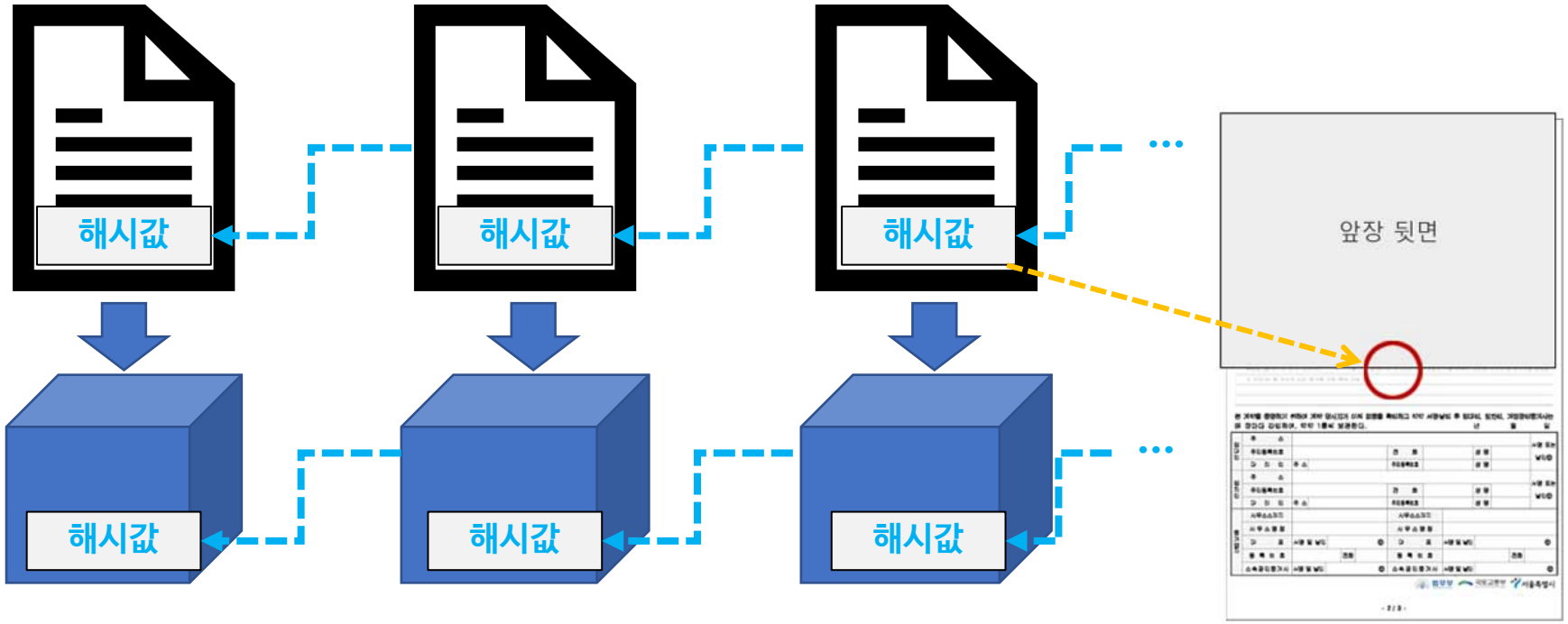
Hello, World? —해시!→ F16C3BB0532537ACD5B2E418F2B1235B29181E35CFFEE7CC29D84DE4A1D62E4D

- 원래는 비밀번호 저장, 문서 위변조 방지에 사용됨



3. 블록체인 기술의 구성요소

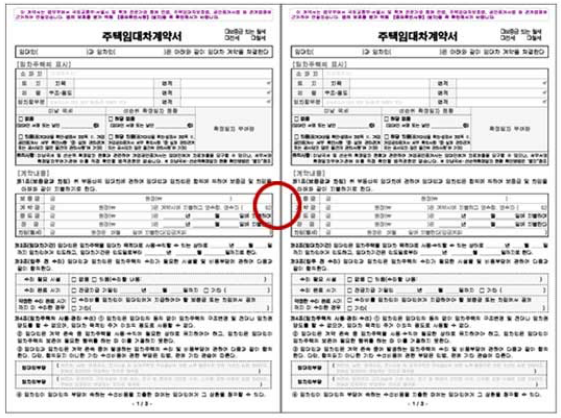
- 간인: 해시 기술 (Hash)
  - 이전 데이터의 해시값을 함께 저장함
  - 저장된 데이터를 위조하려면, 그 뒤에 저장한 모든 데이터를 위조해야 함
    -〉 위변조가 현실적으로 불가능



3. 블록체인 기술의 구성요소

- **복사한 계약서 및 할인: 합의 기술 (Consensus)**

  - 블록체인에서의 합의: **참여자들 사이의 합의**
  - 그럼... 참여자가 몇명?
    - 비트코인 노드 10,005 개 (2019년 기준)
    - 이더리움 노드 7,598 개
  - 만명 도장찍기 =〉 소프트웨어 서비스는 사람의 공간적 한계를 뛰어넘음

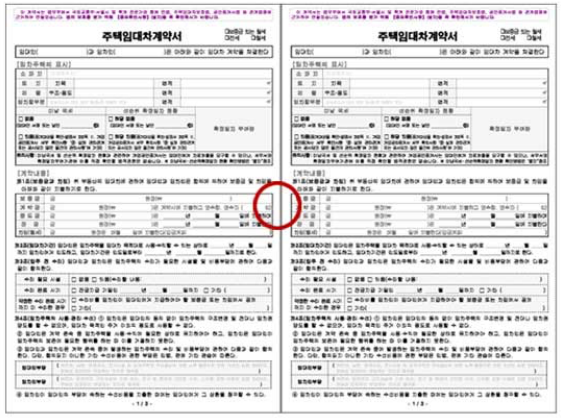


3. 블록체인 기술의 구성요소

- **복사한 계약서 및 할인: 합의 기술 (Consensus)**

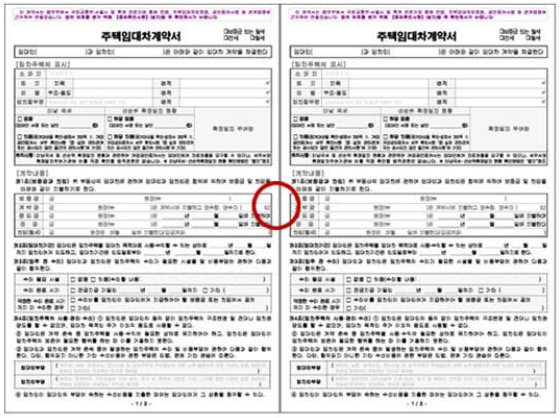
  - 공개 블록체인(Public Blockchain)
    - 블록 생성을 위해서는 문제를 풀어야 함 -〉 조건에 맞는 랜덤 숫자 찾기
      - 맞추면 블록 만들고, 만든 사람에게 보상을 부여 =〉 **"채굴"**
    - 전세계로 퍼져나가는데 오랜 시간이 걸림
      -〉 비트코인, 이더리움이 합의 완료까지 오랜 시간이 걸리는 이유

  - 비공개 블록체인(Private/Permissioned Blockchain)
    - 참여자는 기관별로 제한됨 (보통 2~5개)
    - 은행간 거래, 공공기관 블록체인 등에 활용됨
    - 합의는 보통 만장일치 합의로 이루어지지만,
      참여자 숫자가 적기 때문에 매우 빠름



3. 블록체인 기술의 구성요소

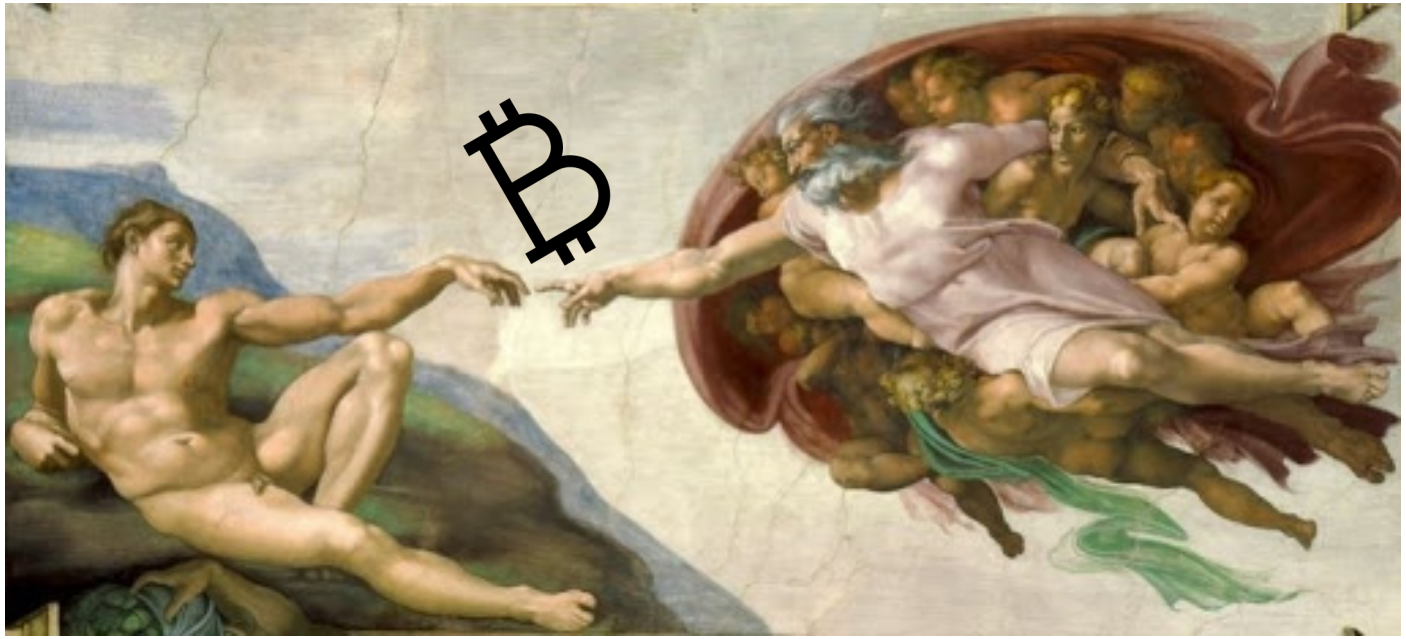
- **복사한 계약서 및 할인: 합의 기술 (Consensus)**
  - 비공개 블록체인(Private/Permissioned Blockchain)
    - 합의 알고리즘
      - Raft: 엄밀히 말해, 비잔틴 장군 문제에 대한 알고리즘 아님
      - PBFT: 현실적 구현이 만만치 않음



3. 블록체인 기술의 구성요소

- **비트코인 (암호화폐)**
  - 블록체인 기술을 이용해서 "**이체 이력**"을 한번 저장해보자
  - 수많은 사람들의 "이체확인증"을 (2019년 기준) **10,005개의 컴퓨터에 저장**해둔 셈

  - 2009년 1월 3일, 사토시 나카모토가 첫번째 블록을 만들고 50 비트코인을 만들면서 시작됨



3. 블록체인 기술의 구성요소

- 인감도장: 공개키/개인키 기술 공인인증서 기술
- 간인: 해시 기술 비밀번호 저장 기술
- 복사한 계약서 및 할인: 합의 기술 네트워크 기술

- 모든 건 이미 있던 기술… 혁신은 아무것도 없는 곳에서 갑자기 오는 것이 아니라, 이미 있던 것들이 조합되어 하나의 서비스가 될 때 이루어짐.
- “내가 그의 이름을 불러주기 전에는 그는 다만 하나의 몸짓에 지나지 않았다. 내가 그의 이름을 불러주었을 때 그는 나에게로 와서 꽃이 되었다” (김춘수, 꽃)

4. 블록체인 기술의 장점

- **위변조 불가**
  - 저장된 데이터는 위변조에 매우 강함
  - 사람이 갖는 공간적, 시간적 한계를 뛰어넘기 때문에, 체인 형태로 구성된 데이터를 모두 위조하는건 "현실적"으로 불가능
  - 공공 목적으로 활용될 때 가장 고려할만한 장점

- **투명성**
  - 모든 참여자들이 데이터를 나누어가짐
  - 지금도 전세계 비트코인 거래 내역은 투명하게 모두가 볼 수 있음
  - 암호화폐 거래를 통해 자신을 숨길 수 있다는 것은 허상
    (자신의 공개키를 아는 순간 전세계 모든 사람이 거래내역을 파악할 수 있음)
  - 프라이버시와 다름을 명확히 이해할 것

5. 블록체인 기술의 한계

- **프라이버시**
  - 사실 블록체인과 관계없음.
  - 프라이버시는 서비스 설계에서 논할 일이지, 블록체인을 쓴다고 해결되는 문제가 아님!
  - 투명성이 장점인 블록체인에 개인정보를 올리는 것은 “매우 섬세한” 서비스 설계가 필요한 일

- **성능**
  - 블록체인은 성능 관점에서 제약이 심한 시스템: 느리고, 자원을 많이 차지함.
  - 민간 기업 도입이 무산되는 가장 큰 이유
  - 투명성, 위변조 불가를 통해 더 상위의 가치를 달성할 경우, 성능 제약에도 도입 가능
    =〉 예: 공공 영역 (투표, 문서 위변조 방지 등)

프라이빗 블록체인의 현재

2020-10-19

빅픽처랩(주)

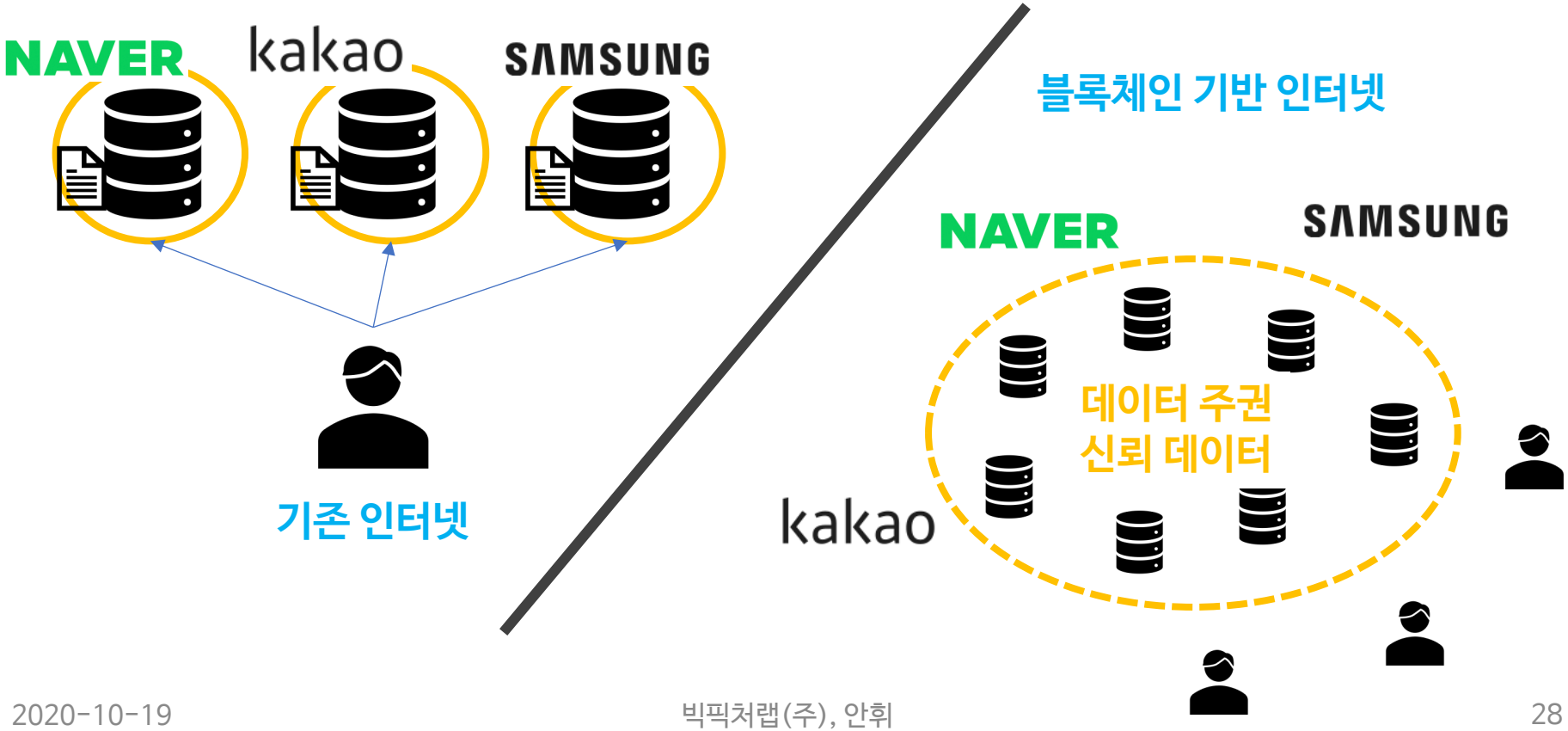
안휘

목차

1. 블록체인 기술의 비전
2. 블록체인 기술의 현주소
3. 탈중앙화 신원인증
4. 지역화폐
5. 중앙은행발행 디지털 화폐

1. 블록체인 기술의 비전

- 인터넷에 신뢰할 수 있고 민주적인 데이터 레이어의 생성
  =〉 인터넷 데이터 관점의 민주화, 세계화



1. 블록체인 기술의 비전

- **인터넷에 신뢰할 수 있고 민주적인 데이터 레이어의 생성**
  =〉 인터넷 데이터 관점의 민주화, 세계화

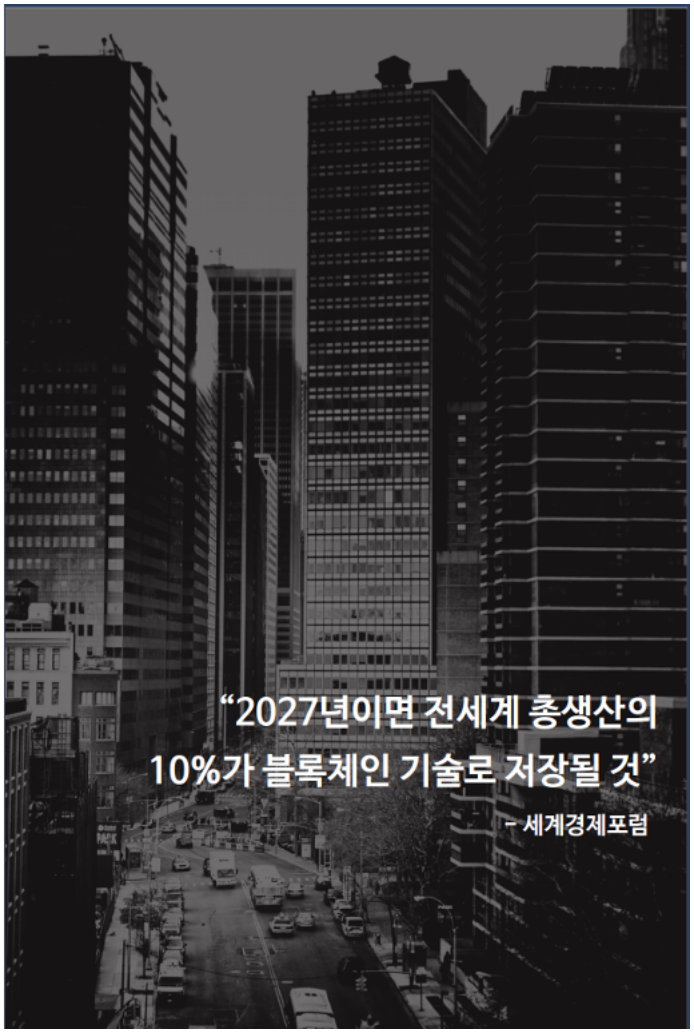
- **기존 인터넷**
  - 데이터는 각 개별 회사/기관의 것
  - 개인 데이터임에도 데이터 자체는 회사가 소유, 관리, 판매 등 영업 활용
  - 불투명성이 매우 큰 문제 =〉 최근 페이스북 등이 큰 문제를 일으킴

- **블록체인**
  - 데이터의 투명성
  - 데이터의 권리를 디지털 인감도장(PKI) 기술 기반으로 개인에게 이전

이 비전을 가장 중앙 집권화된 시스템인 화폐 시스템에 적용한 것이
비트코인으로 대표되는 암호화폐

1. 블록체인 기술의 비전

INSIDE: A 12-PAGE SPECIAL REPORT ON COLOMBIA
The Economist
Our guide to America's best colleges
Myanmar's free-ish election
Those ever-creative accountants
America takes the fight to IS
Coywolves: the new superpredator
Economist.com
The trust machine
How the technology behind bitcoin could change the world
IN BLOCKCHAIN WE TRUST


"2027년이면 전세계 총생산의
10%가 블록체인 기술로 저장될 것"
- 세계경제포럼

2. 블록체인 기술의 현주소

2. 블록체인 기술의 현주소

- **암호화폐의 모습에 대한 실망감**
  - 화폐 시스템이 현재와 같은 강력한 통제를 갖게 된 것에는 다 이유가 있음
  - 화폐, 주식, 부동산 등의 자산과는 다르게 암호화폐 자체는 실물 기반이 없음
  - 거래 내역 자체가 실물 기반 역할을 함
- **그럼에도 혁신은 지속됨**
  - 인터넷에 신뢰 데이터 레이어를 구축하고자 하는 이더리움
  - 킬링 어플리케이션의 부재가 지속적인 발목을 잡음
  - 기술자 커뮤니티에서는 지속적으로 사용성에 대한 가능성을 타진 중
- **아직은 초기 기술로 지속적인 투자 연구가 필요**
  - 기술 컨샙과 가능성은 매우 훌륭함
  - 아직 시장을 장악한 주도적인 소프트웨어 시스템의 부재로 가능성이 매우 넓게 열려 있음
  - **투표**, **공문서 저장**, **유통망 추적** 등에서 유의미한 성과를 보이고 있음 =〉 공공영역에서의 가능성이 높아지고 있음

2. 블록체인 기술의 현주소

- 탈중앙화 신원증명 (a.k.a DID)
- 지역화폐
- 중앙은행발행 디지털 화폐

3. 탈중앙화 신원증명

- 탈중앙화 신원증명(DID, Decentralized IDentity)
  - "신분증"의 "인증수단"을 저장하는 컨셉

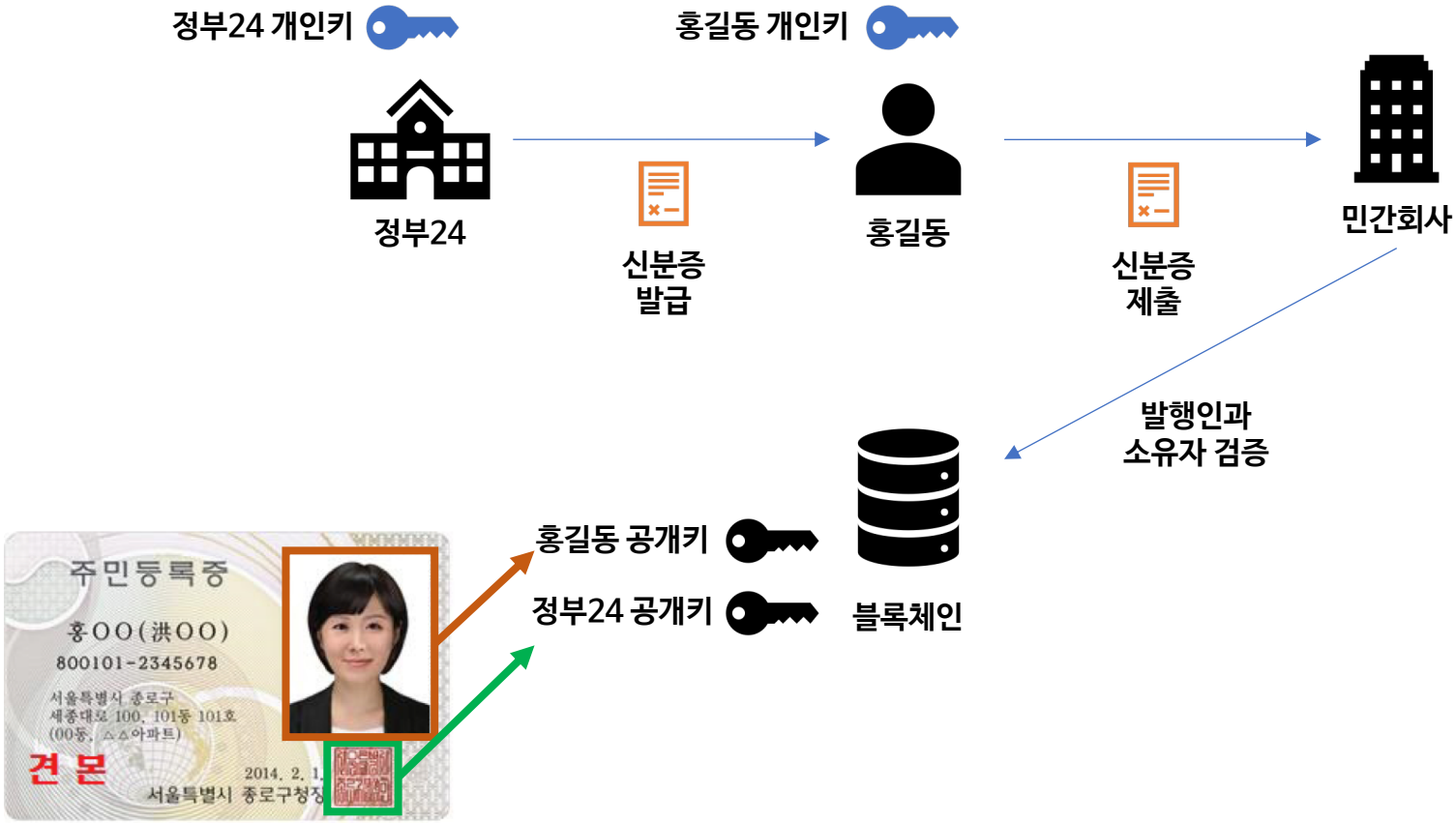
디지털 인감도장 기술
: 공개키/개인키 기술 사용

신분증의 구조
(출처: 자기주권 신원증명 구조분석서(제이펍), 윤대근)

- 요즘 가장 떠오르는 블록체인 활용 분야
- 해외는 물론 국내에서도 아주 활발히 논의됨

3. 탈중앙화 신원증명

- 탈중앙화 신원증명(DID, Decentralized IDentity)



3. 탈중앙화 신원증명

- **Pass 앱 운전면허증**
  - 주요 개인정보는 사용자 앱에 저장하고 블록체인에는 최소한의 식별을 위한 정보만 저장
  - 전산화 함으로써 얻는 사용상 편의성 및 효율성을 추구하면서 위변조를 방지하는 것이 매우 중요 =〉 블록체인 기반 소프트웨어 서비스 구축

4. 지역화폐

- **현재 블록체인 기반 지역화폐**
  - 노원구 지역화폐 NW
  - 조폐공사 지역화폐 플랫폼 Chak

대한민국 신뢰플랫폼 chak

chak
Change for Korea

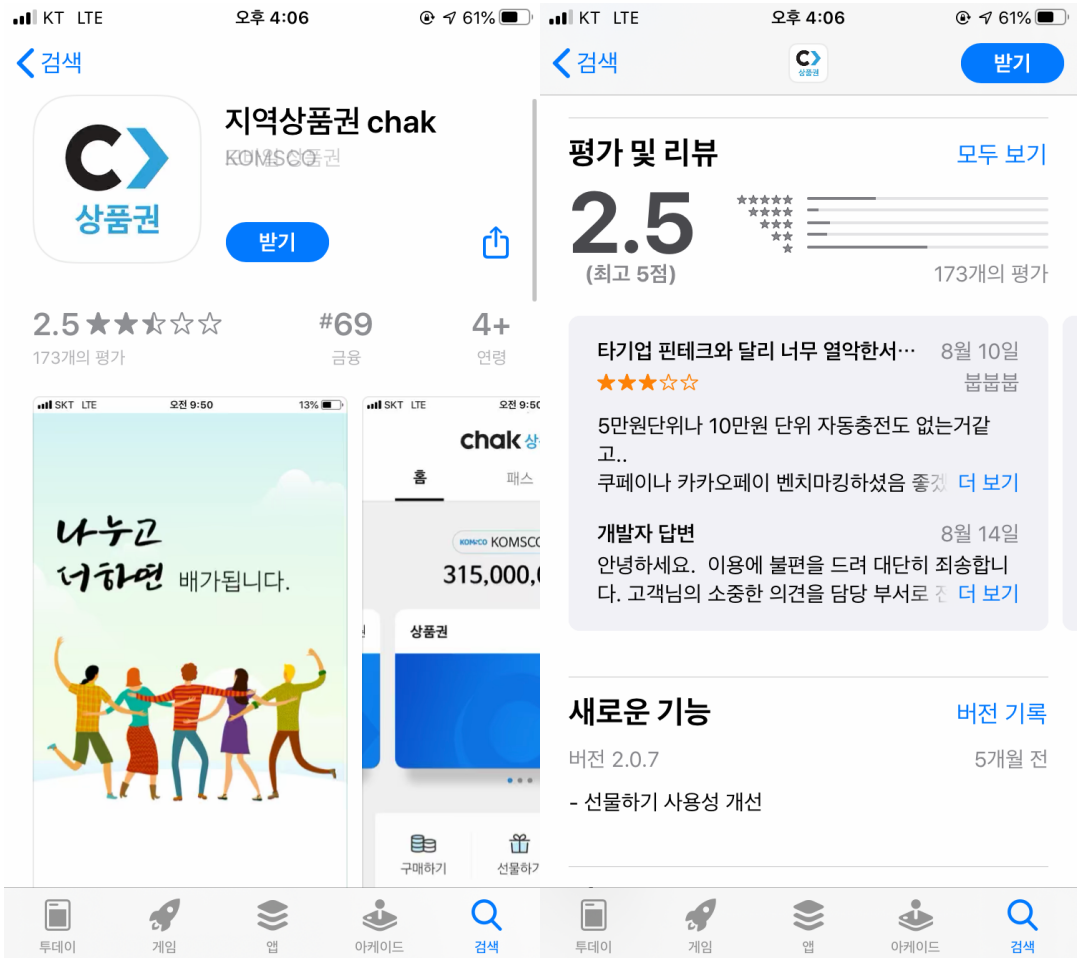
대한민국의 새로운 미래와 내일을 위한 즐거운 변화!

그동안 번거롭고 귀찮았던 결제·인증·보안 서비스를 착착! 해결해주고, 생활에 착! 달라붙어 대한민국 국민들의 삶을 보다 편리하게 만드는 국민 생활 밀착! 플랫폼

  - 종이, 상품권 등으로 관리되던 지역화폐의 전산화 시도
    - "블록체인": 암호화폐 등을 통해 이미 화폐 플랫폼 기술로 검증
    - **"전산화" 자체에 촛점**이 맞춰질 필요는 있음
      - 블록체인을 쓰는게 중요한 것이 아님
      - 중요한 건 좋은 "서비스"를 제공하는 것

4. 지역화폐

- **현재 블록체인 기반 지역화폐**
  - 노원구 지역화폐 NW
  - 조폐공사 지역화폐 플랫폼 Chak



4. 지역화폐

- **지역화폐를 블록체인 기반 시스템으로 만들 당위성**
  - 블록체인에 저장해도 좋을 데이터의 AND 조건 검사
    - 거래 내역은 위변조가 되어선 안됨
    - 거래 내역의 보관은 참여 지자체, 기관, 시민단체 등이 모두 가져야 함
      - 단순히 지자체가 데이터를 모두 갖기 보단 외부 기관, 단체의 감사가 있을 때 더 완전해질 수 있음

  - 블록체인에 저장하면 안될 데이터의 OR 조건 검사
    - 실시간성: ms 단위의 엄밀한 실시간성을 요구하지는 않음
      (거래 자체는 1초 내외로 이루어지면 됨 → 제대로 이루어지는 것이 더 중요)
    - Query는 매우 단순함: 잔액 계산을 위한 검색 정도만 필요
    - 참여 기관 사이에서 공유되어서는 안 될 개인 데이터 아님

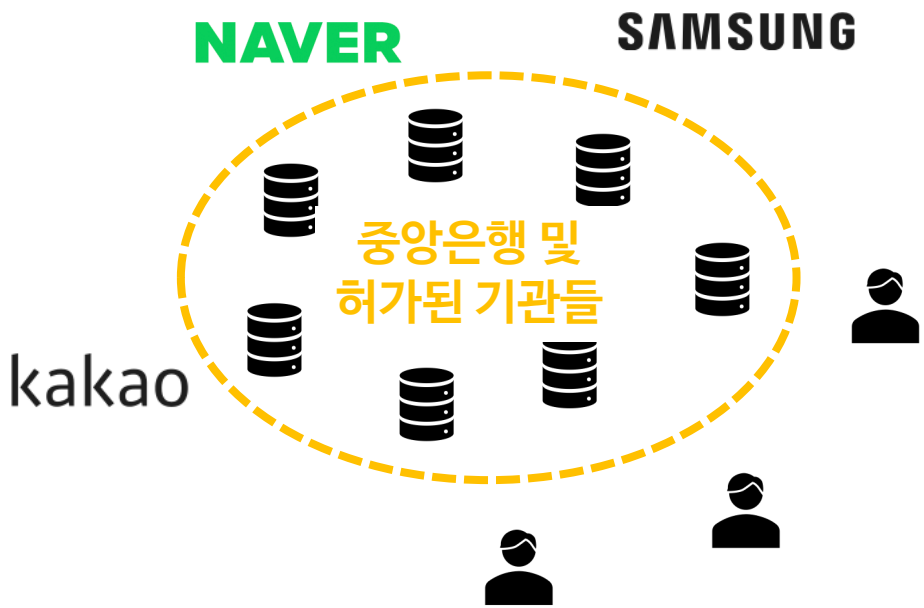
4. 지역화폐

- **해결해야 할 기술적 문제**
  - 첫째도, 둘째도 사용 가능한 수준의 실시간성 확보
    - MVCC, Phantom Conflict 해결 → 해결 못하면 여기서 끝.
    - 고성능의 일반 데이터베이스를 이용한 캐싱을 적극적으로 활용
      → 블록체인은 **Source of Truth**, 서비스는 서비스를 위한 DB에서…

- **확장 가능성**
  - DID 기술을 활용하여 지역 주민 인증 등과 결합
  - 공공기관의 협조 필요

5. 중앙은행 발행 디지털 화폐

- 이체내역을 저장하는 컨샙은 암호화폐와 동일
- 중앙은행 및 허가된 기관들(공공기관들, 은행들)이 함께 데이터를 관리/운영



5. 중앙은행 발행 디지털 화폐

- **스웨덴 중앙은행 (e-krona (이크로나))**
  - 현금사용의 급격한 감소
  - 현금과 같은 공공재 성격의 지급수단 수요 증가
  - 민간 전자지급수단에 대한 의존도 심화로 나타나는 지급서비스시장 독점 문제
    - 전자지급수단: 스마트폰 뱅킹, xx페이 앱 등

- **중국 인민은행**
  - 민간 디지털화폐로 인한 위안화의 국제적 지위 훼손 방어 차원

- **한국은행**
  - 최근 2019년 이후부터 CBDC에 대한 각종 보고서를 내놓기 시작
  - CBDC에 대해 진지하게 계획을 하는 것으로 보임

무엇보다 중요한 것은,
"어떠한 기술을 사용하느냐"가 아니라,
사용자에게 맞는 "좋은 서비스를 제공"하는 것이고,
이 대명제를 지키기 위한 방안으로 "특정 소프트웨어 기술"을 활용하는 것
블록체인이라고 다르지 않음.

하이퍼레저 패브릭을 구성하는 기술들

2020-10-19

빅픽처랩 ㈜

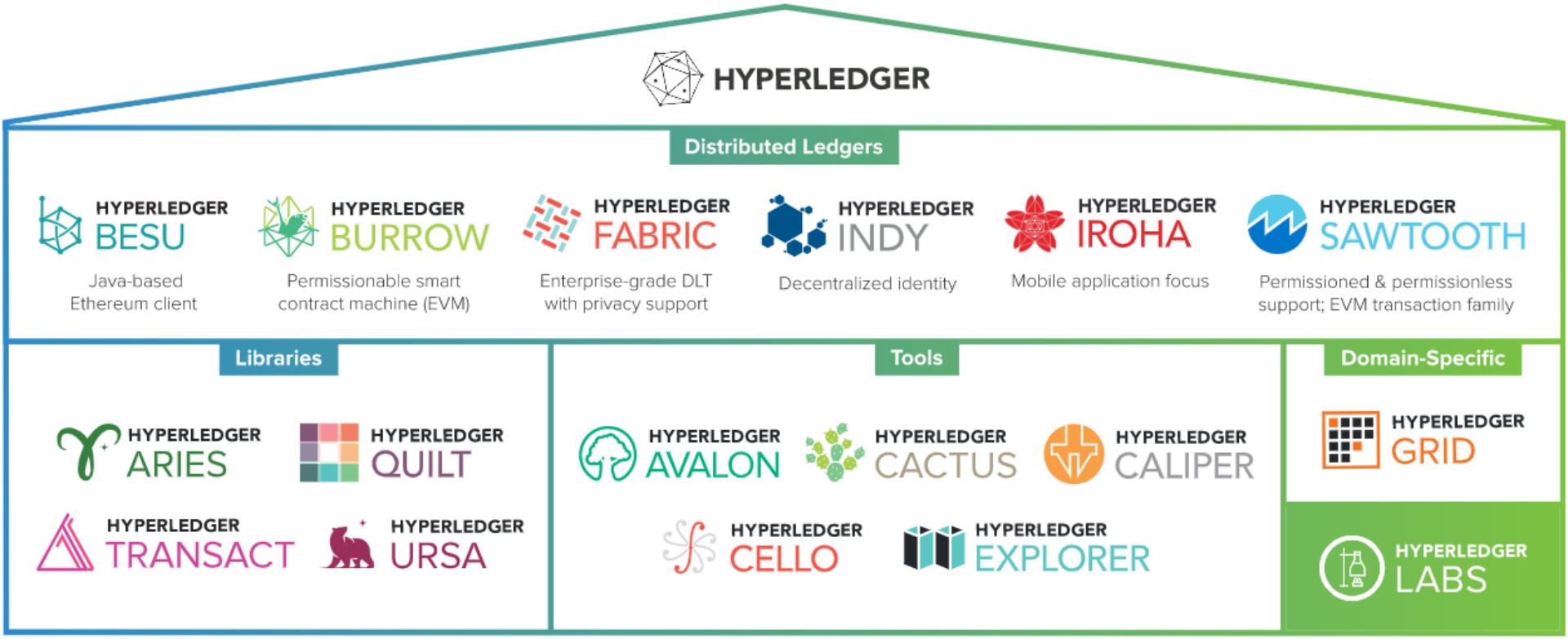
안 휘

목차

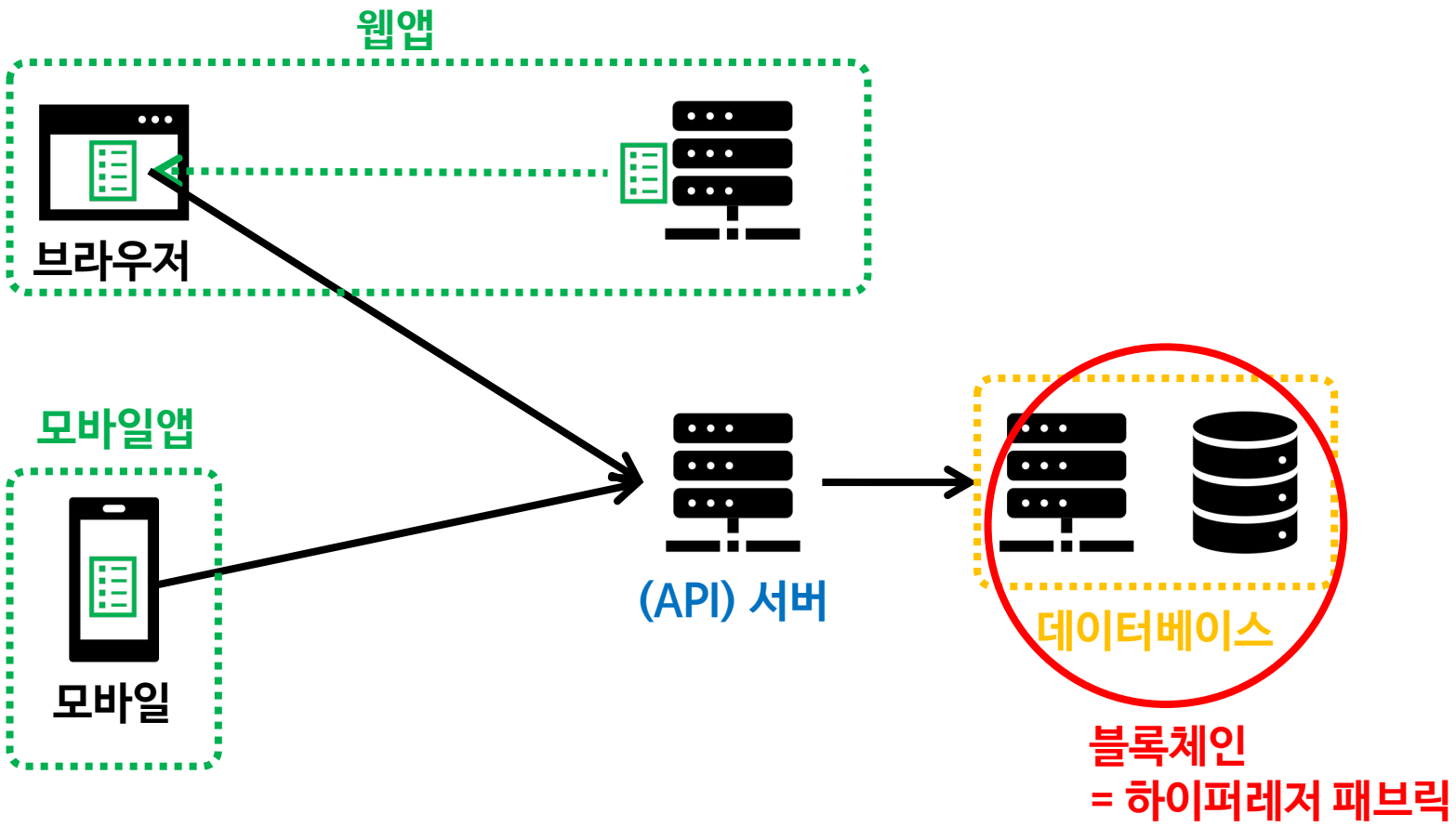
1. 하이퍼레저
2. 하이퍼레저 패브릭
3. 하이퍼레저 패브릭 구조
4. 하이퍼레저 패브릭을 구성하는 기술들

1. 하이퍼레저

- "Hyperledger is an open source community focused on developing a suite of stable frameworks, tools and libraries for enterprise-grade blockchain deployments."



2. 하이퍼레저 패브릭



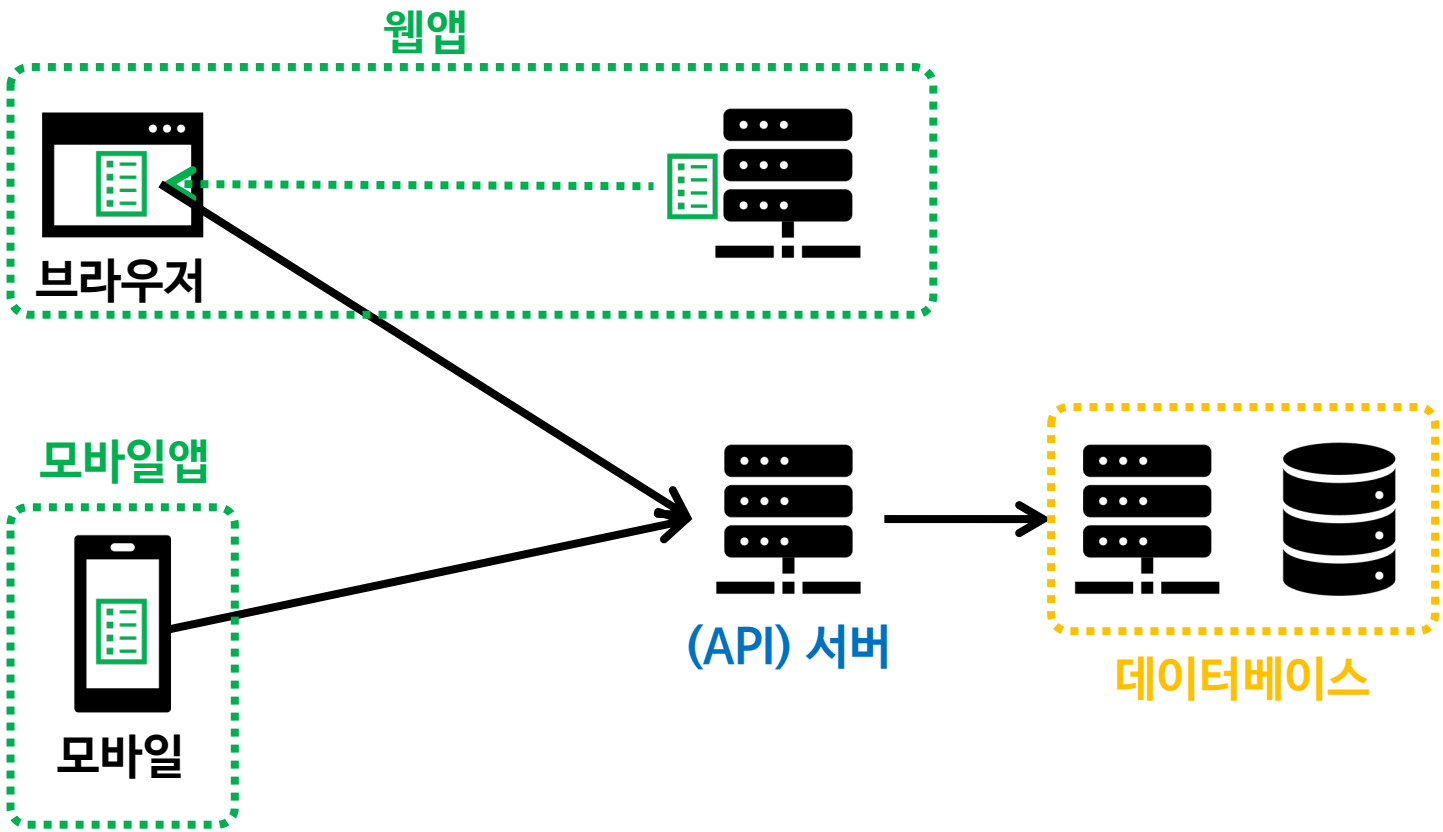
2. 하이퍼레저 패브릭

- "Hyperledger Fabric is an open source enterprise-grade permissioned distributed ledger technology (DLT) platform, designed for use in enterprise contexts, that delivers some key differentiating capabilities over other popular distributed ledger or blockchain platforms."
- 하이퍼레저 안에서 가장 먼저 1.0 버전을 론칭 (2017.07)
  - 1.4.0 (첫 LTS) 버전: 2019.01
  - 2.0 버전: 2020.01
  - 2.2 (LTS) 버전: 2020.7

2. 하이퍼레저 패브릭

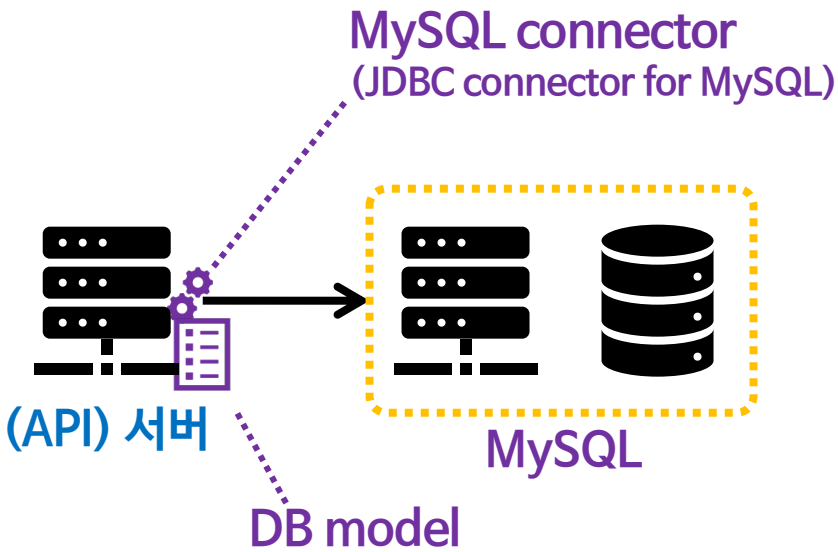
- Highly modular & configurable
  - Enterprise의 다양한 요구에 대응하기 위함
  - 하이퍼레저 패브릭을 복잡하게 하는 주 원인
- Java, Go, JavaScript 같은 범용 언어로 스마트컨트랙트 작성이 가능
- Permissioned
- Pluggable consensus protocol
  - 큰 의미 없고, RAFT 라고 생각하면 됨
- NOT require native cryptocurrency
  - Ethereum 프라이빗 버전을 쓰지 않고 하이퍼레저 패브릭을 선택하는 주된 이유

3. 하이퍼레저 패브릭 구조

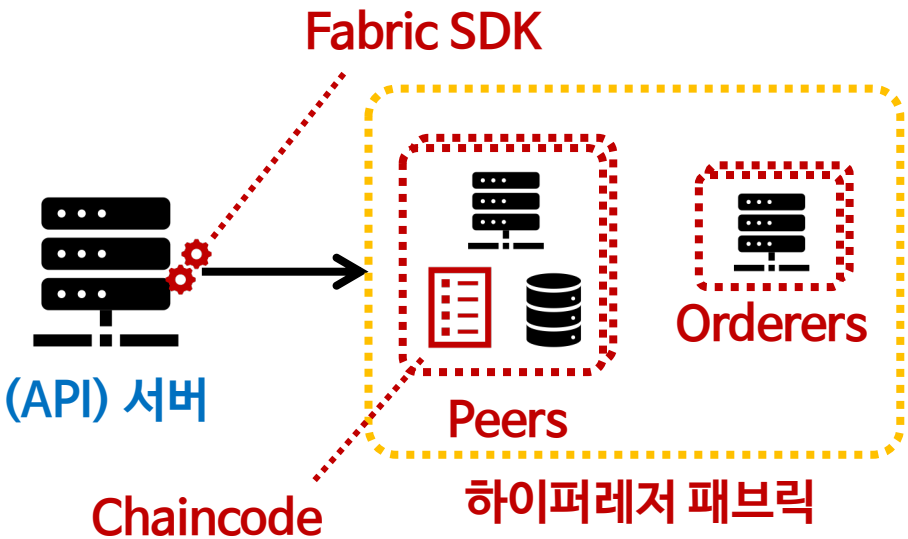


3. 하이퍼레저 패브릭 구조

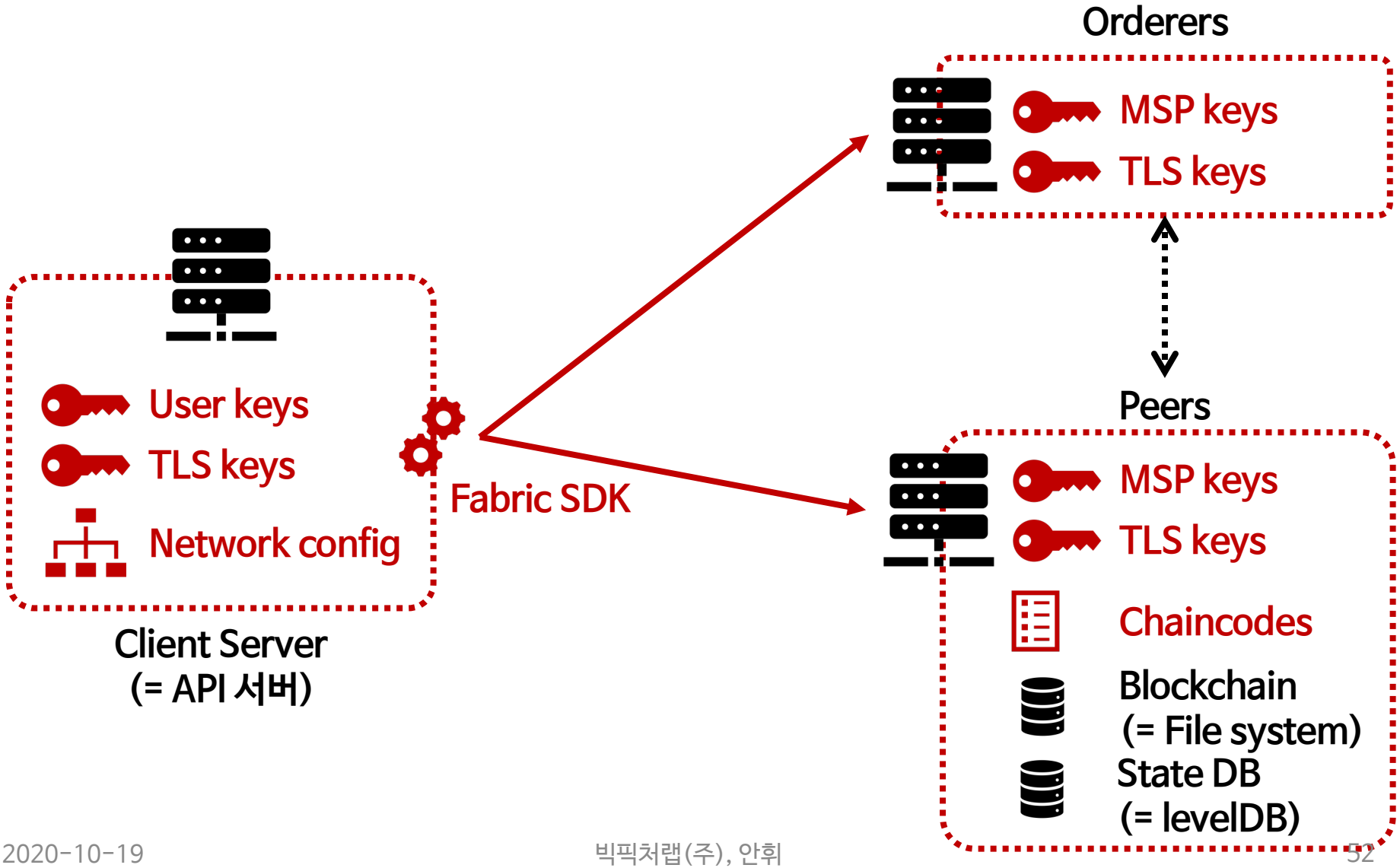
일반적인 Server – DB 구조



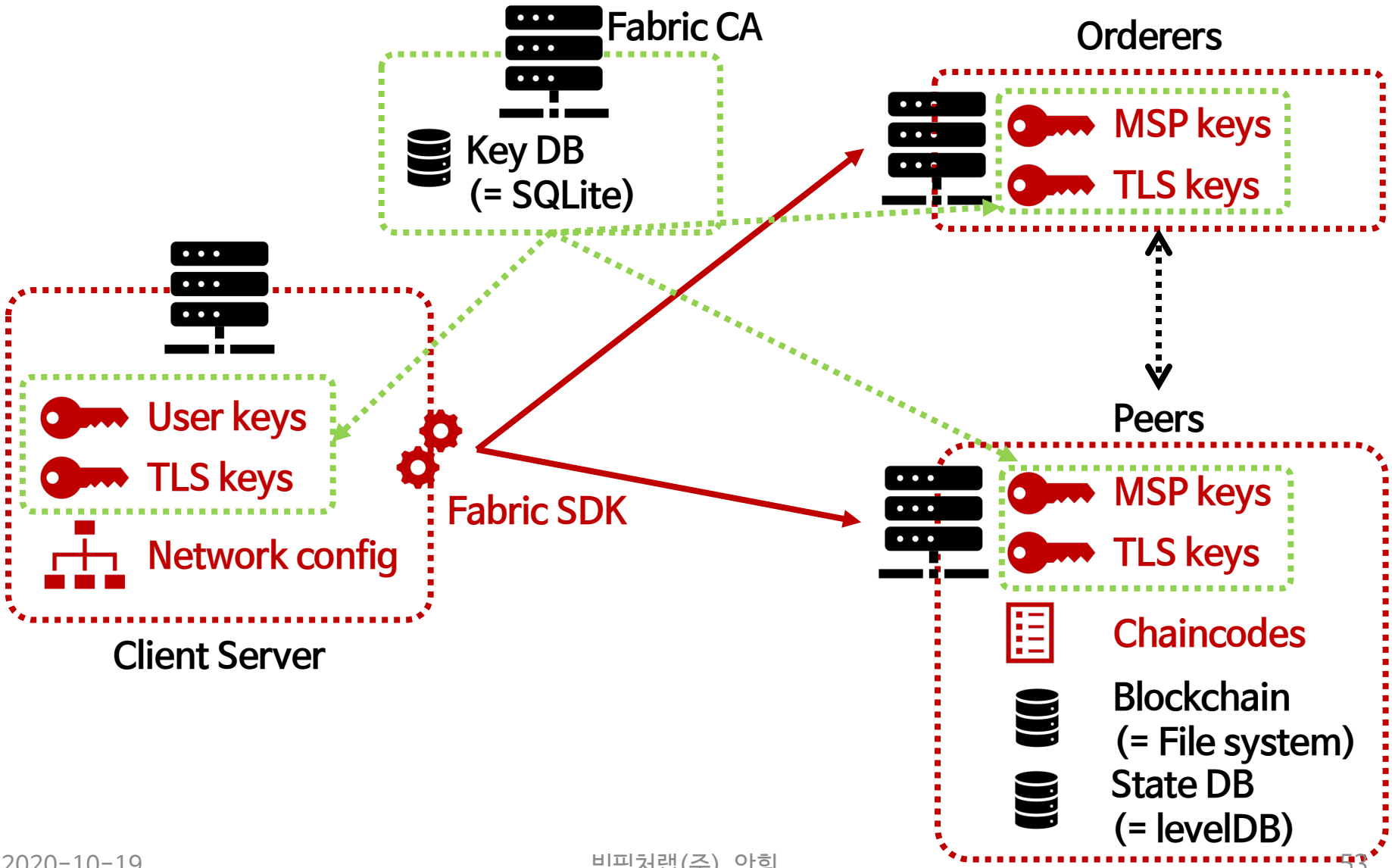
하이퍼레저 패브릭 Server – DB 구조



3. 하이퍼레저 패브릭 구조



3. 하이퍼레저 패브릭 구조



4. 하이퍼레저 패브릭을 구성하는 기술들

- **Docker container**
  - Orderer, Peer, Fabric CA 모두 Docker Image로 배포됨
  - Binary도 있기 때문에 그냥 실행도 가능
- **Unix environment**
  - Linux 또는 macOS 환경이 필요
  - Windows도 가능하지만, 제공되는 기본 스크립트들이 모두 Unix shell 환경을 기준으로 작성됨
- **프로그래밍 언어**
  - Go: 하이퍼레저 패브릭, 체인코드, Fabric SDK (beta)
  - JavaScript(node.js): 체인코드, Fabric SDK
  - Java: 체인코드, Fabric SDK

4. 하이퍼레저 패브릭을 구성하는 기술들

- Docker container => docker, docker-compose
  - Orderer, Peer, Fabric CA 모두 Docker Image로 배포됨
  - Binary도 있기 때문에 그냥 실행도 가능
- Unix environment => WSL 2
  - Linux 또는 macOS 환경이 필요
  - Windows도 가능하지만, 제공되는 기본 스크립트들이 모두 Unix shell 환경을 기준으로 작성됨
- 프로그래밍 언어
  - Go: 하이퍼레저 패브릭, 체인코드, Fabric SDK (beta)
  - JavaScript(node.js): 체인코드, Fabric SDK
  - Java: 체인코드, Fabric SDK

하이퍼레저 패브릭 구동 실습

2020-10-19

빅픽처랩㈜

안 휘

실습 목차

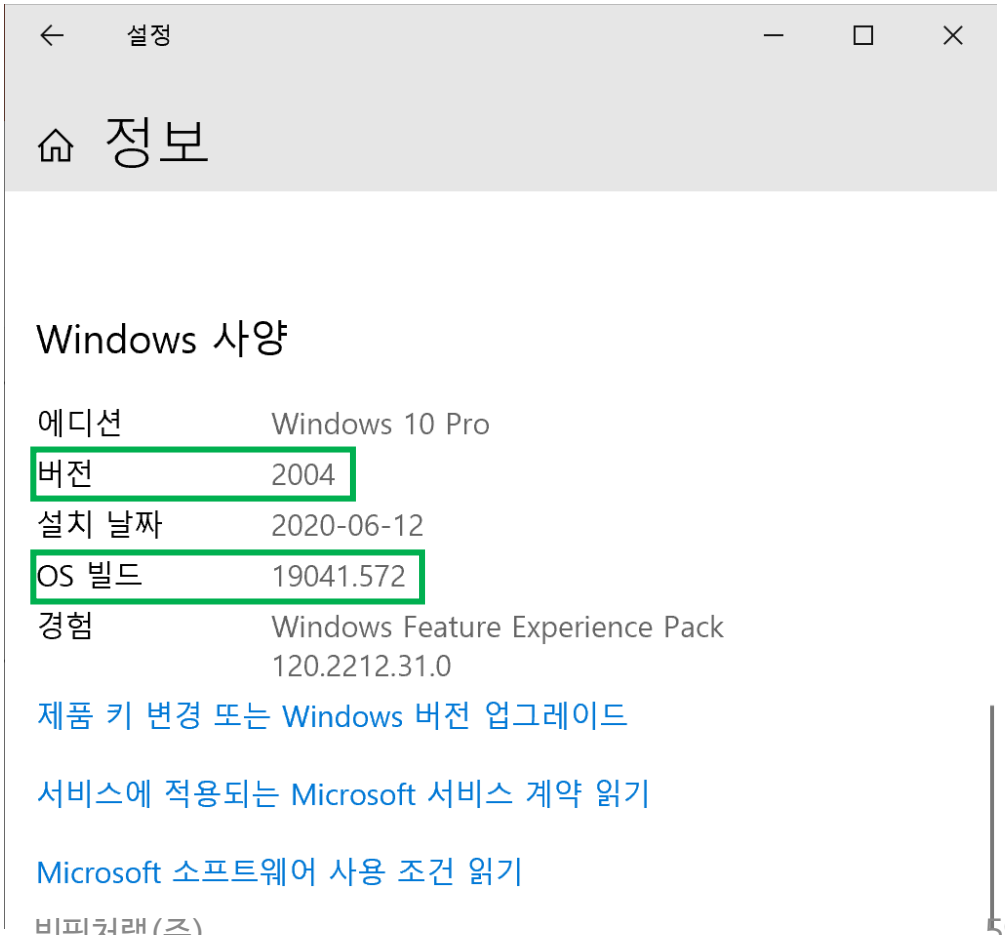
- **실행 환경 구성 (1시간)**
  - WSL 2 설치
  - Visual Studio Code 설치
  - docker, docker-compose 설치
- **하이퍼레저 패브릭 설치 (1시간)**
  - 하이퍼레저 패브릭 설치
  - test-network 실행
- **하이퍼레저 패브릭 실행 (1시간)**
  - 채널 생성
  - 체인코드 패키징 및 배포
- **하이퍼레저 패브릭 Client Server 실행 (1시간)**
  - node.js 설치
  - Client Server 실행

실행 환경 구성

- WSL 2 설치
- Visual Studio Code 설치
- docker, docker-compose 설치

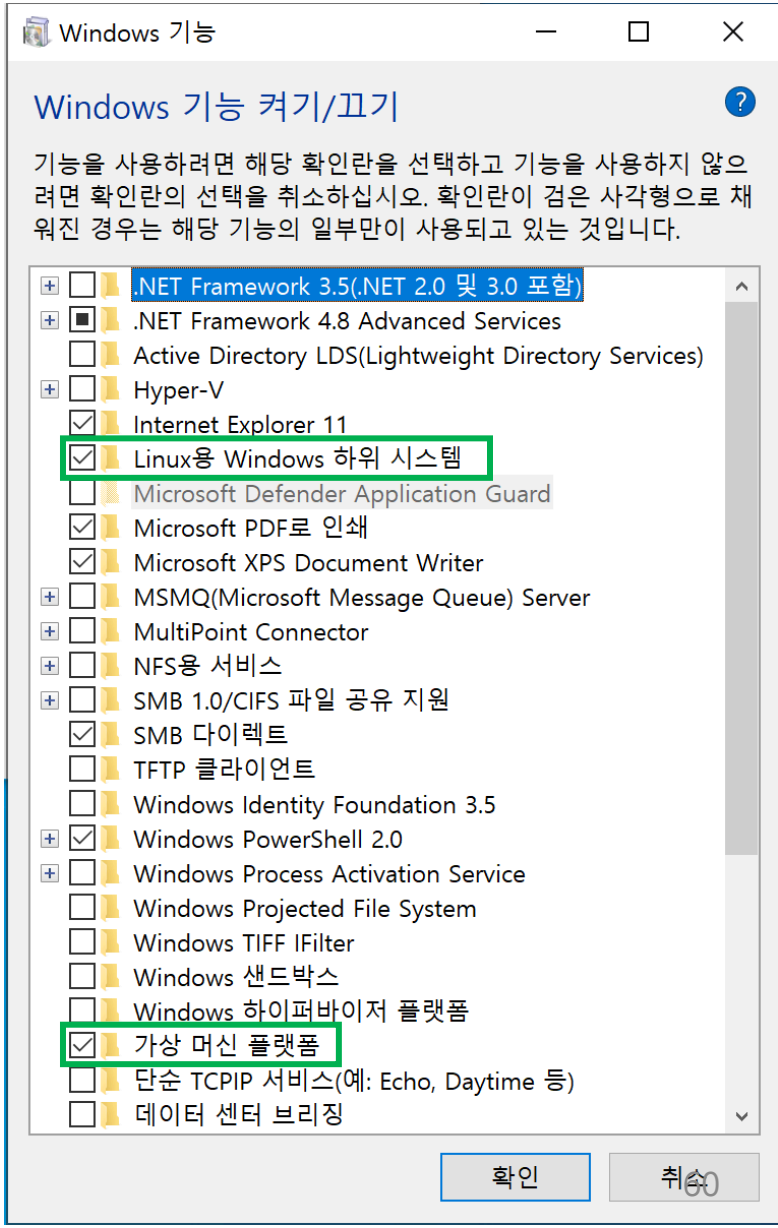
WSL 2 설치

- **Windows 10의 경우 WSL 2를 이용하여 Linux를 사용 가능**
  - 버전 1903 (빌드 18362) 이상 필요
  - “설정” → “정보”



WSL 2 설치

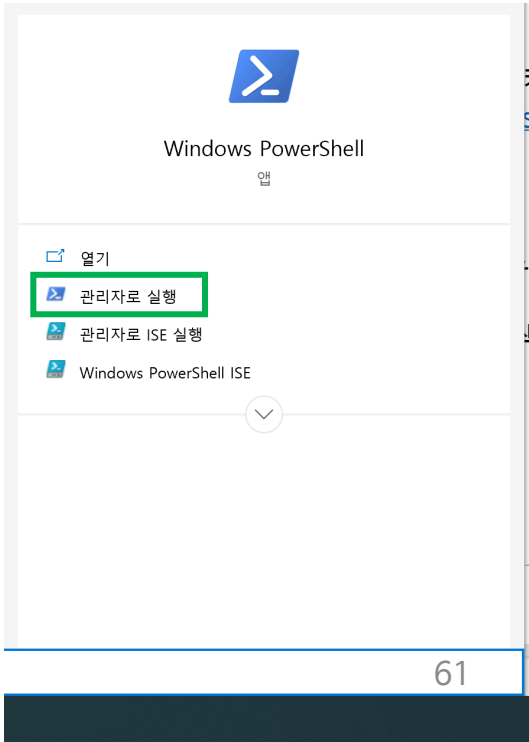
- **Windows 10 WSL 2 설치**
  - Windows 검색
  - “제어판” 검색
  - “프로그램” 선택
  - “프로그램 및 기능” 섹션의 “Windows 기능 켜기/끄기” 선택
  - “Linux용 Windows 하위 시스템”, “가상 머신 플랫폼” 선택
  - “확인”
  - 시스템 다시 시작



WSL 2 설치

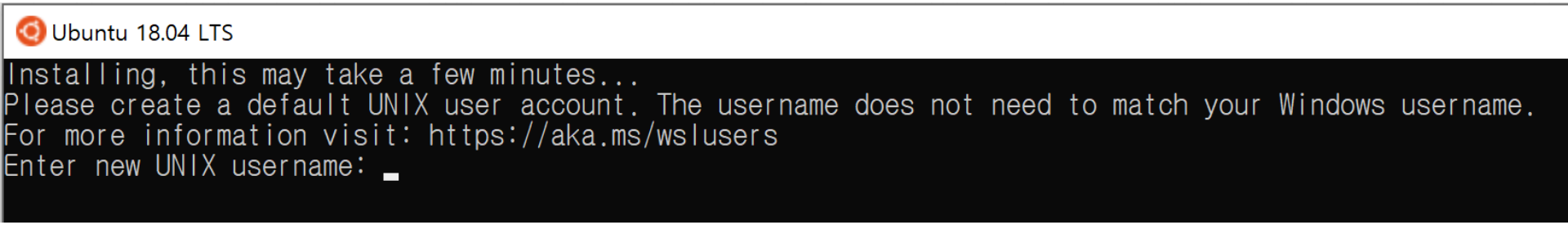
- **Windows 10 WSL 2 설치**
  - Linux 커널 업데이트 패키지 다운로드
    - https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
      - 구글에서 “wsl 설치” 검색 -〉 마이크로소프트 공식 문서 “4단계” 섹션 참조
    - 설치

  - WSL 2를 기본 버전으로 설정
    - Windows 검색
    - “PowerShell” 검색
    - “관리자로 실행” 선택
    - 다음 커맨드 입력
      - wsl --set-default-version 2



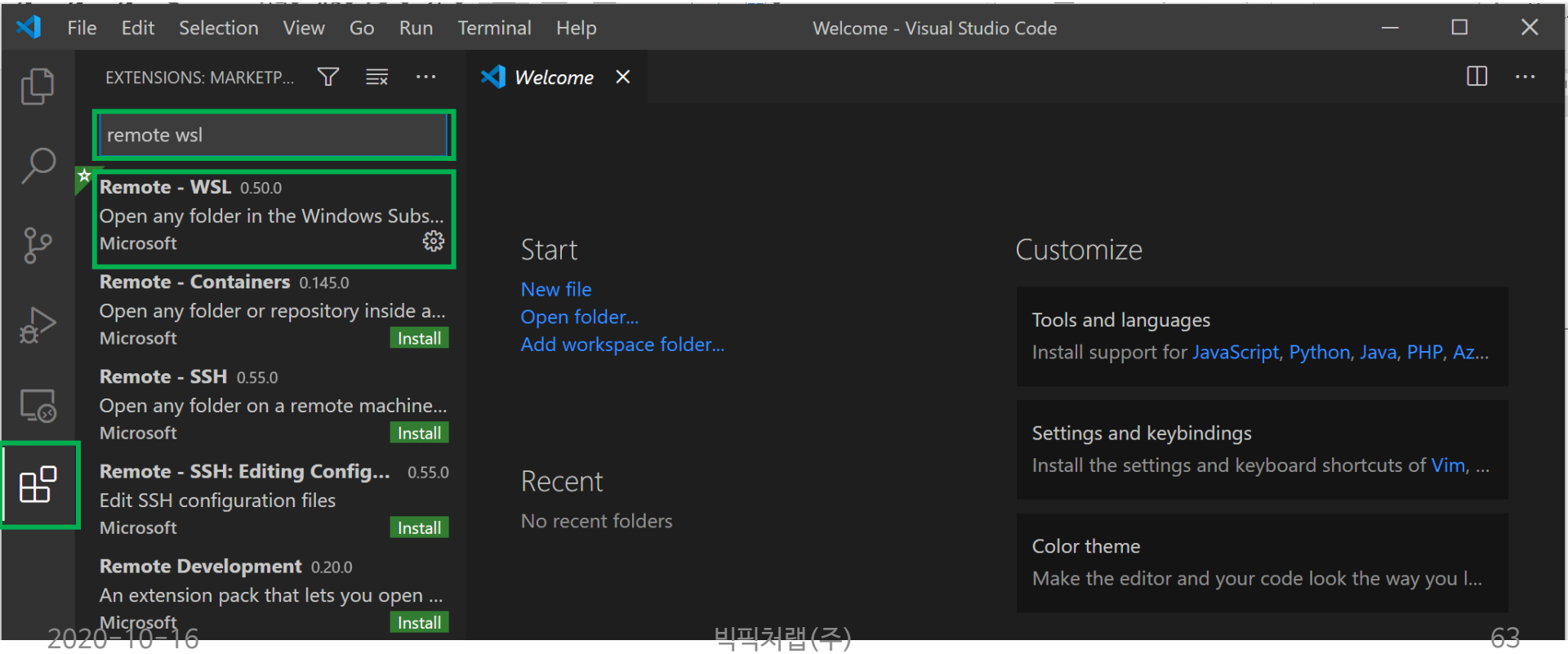
WSL 2 설치

- **Windows 10 WSL 2 설치**
  - Linux 배포판 설치
    - Windows 검색
    - “Microsoft Store” 검색
    - “Ubuntu” 검색
    - “Ubuntu 18.04” 설치
      - Microsoft 로그인 필요 없음
    - 다운로드 완료 후 실행
    - 사용자 계정 생성



Visual Studio Code 설치

- https://code.visualstudio.com/
- 다운로드, 설치 및 실행
- (Windows 10) WSL 접속 확장 프로그램 설치



File Edit Selection View Go Run Terminal Help

Welcome - Visual Studio Code

Welcome

Start

New file
Open folder...
Add workspace folder...

Recent

No recent folders

Help

Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter

Show welcome page on startup

Customize

Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more

Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and ...

Color theme
Make the editor and your code look the way you love

Learn

Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl...

Interface overview
Get a visual overlay highlighting the major components of the UI

Int
Try

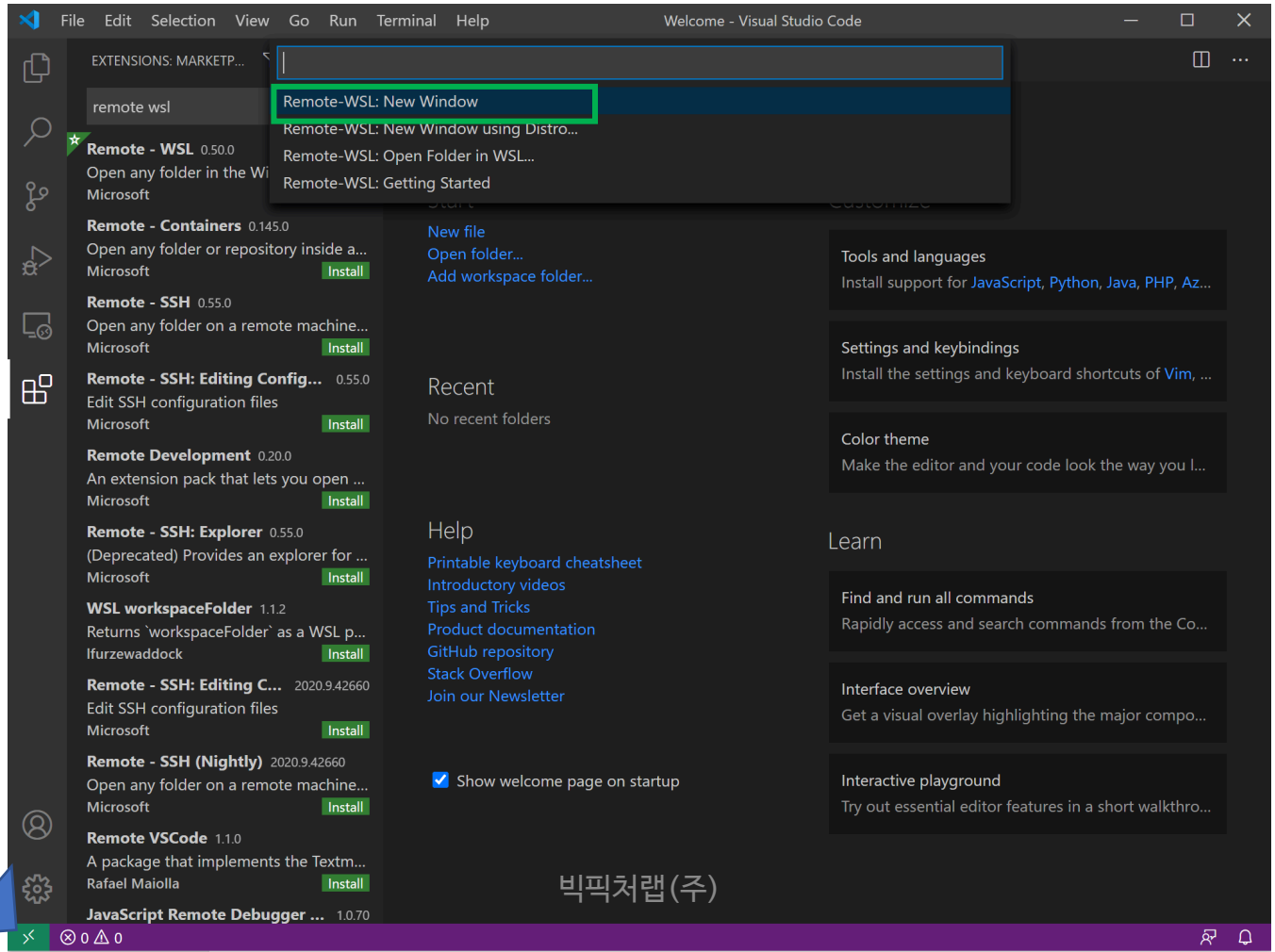
You have Windows Subsystem for Linux (WSL) installed on your system. Do you want to install the recommended extensions for it?

Install | Show Recommendations

⊗ 0 ⚠ 0

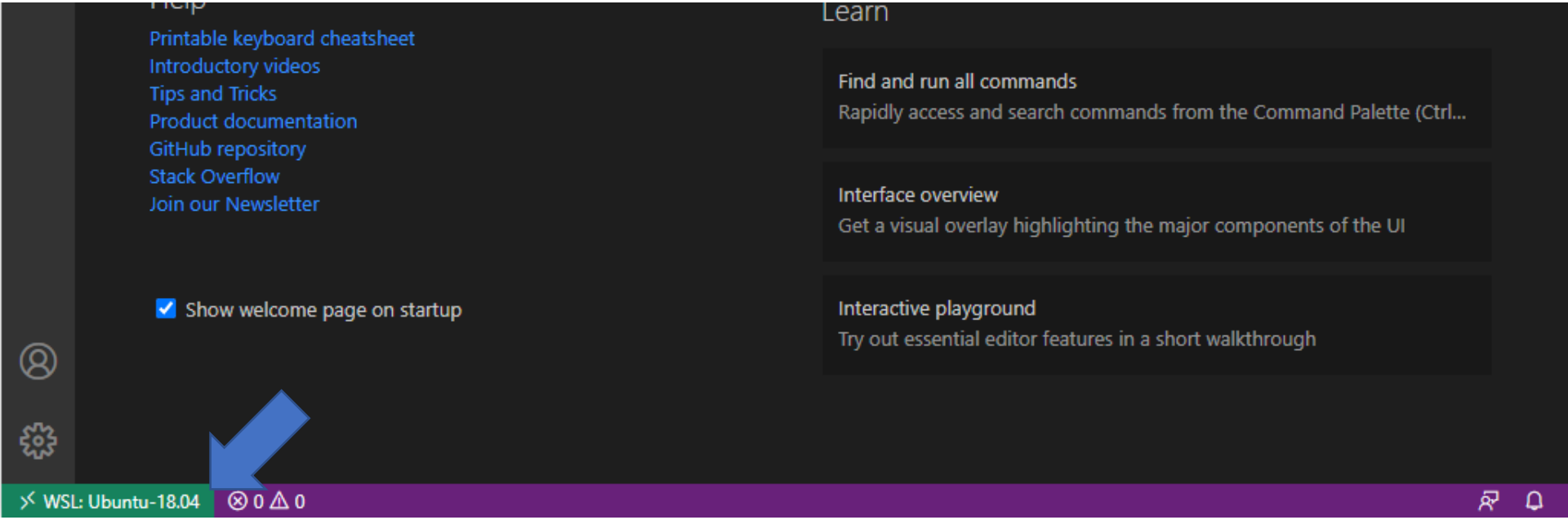
Visual Studio Code 설치

- (Windows 10) WSL 접속 확장 프로그램 설치



Visual Studio Code 설치

- (Windows 10) WSL 접속 확장 프로그램 설치



docker, docker-compose 설치

- https://docs.docker.com/get-docker/
  - Windows, macOS에 맞춰서 다운로드 후 설치
  - Windows의 경우 설치 중 나오는 WSL Integration 옵션 체크 (이미 선택되어 있음)

Get Docker

Estimated reading time: 1 minute

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.

Docker Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.

Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

하이퍼레저 패브릭 설치

- 하이퍼레저 패브릭 설치
- test-network 실행

하이퍼레저 패브릭 설치

- **Ubuntu 를 실행**
- **다음 커맨드 입력**
  - curl -sSL https://bit.ly/2ysbOFE | bash -s
- **명령어를 실행한 곳으로부터 "./fabric-samples" 폴더에 복사됨**
  - 버전은 최신 버전인 2.2.1이 설치됨

test-network 실행

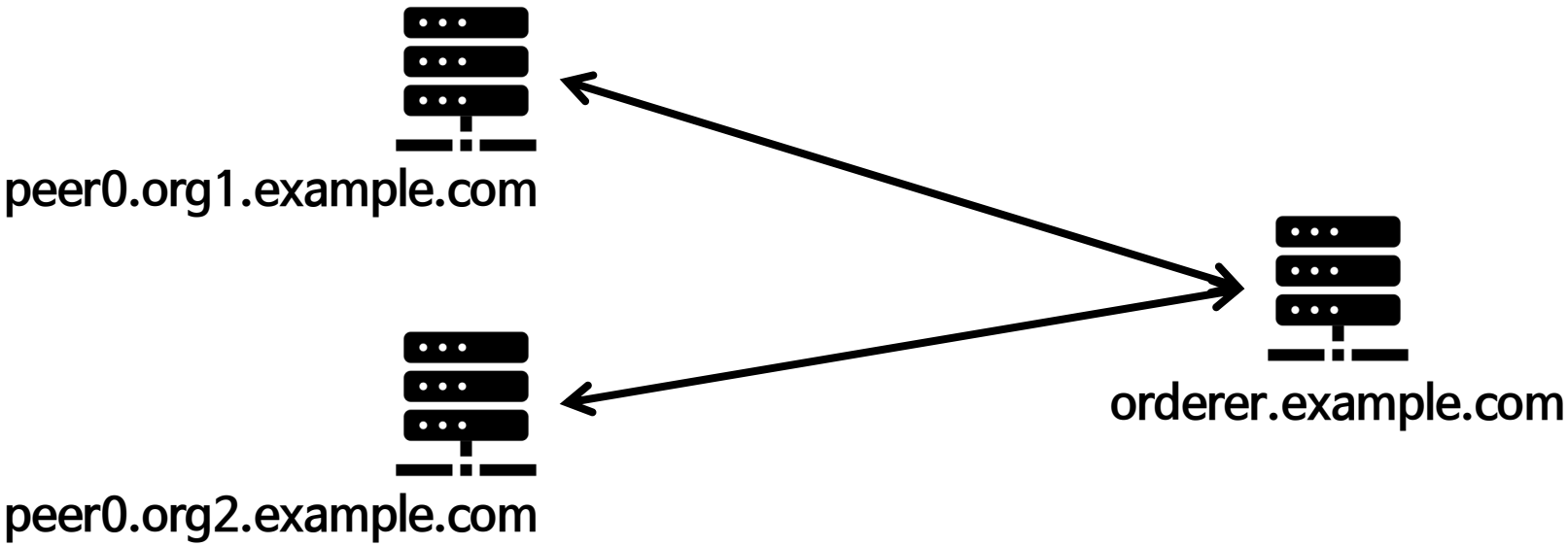
- Ubuntu 를 실행
- 다음 커맨드 입력
  - cd ~/fabric-samples/test-network
- 다음 커맨드 입력
  - ./network.sh up

```
byron1st@DESKTOP-D240NBD:~/fabric-samples/test-network$ ./network.sh up
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=2.2.1
DOCKER_IMAGE_VERSION=2.2.1
/home/byron1st/fabric-samples/test-network/../bin/cryptogen
Generate certificates using cryptogen tool
Create Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Create Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Create Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generate CCP files for Org1 and Org2
/home/byron1st/fabric-samples/test-network/../bin/configtxgen
Generating Orderer Genesis block
+ configtxgen -profile TwoOrgsOrdererGenesis -channelID system-channel -outputBlock ./system-genesis-block/genesis.block
2020-10-18 15:55:15.362 KST [common.tools.configtxgen] main -> INFO 001 Loading configuration
2020-10-18 15:55:15.384 KST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2020-10-18 15:55:15.384 KST [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms" election_ti
ight_blocks:5 snapshot_interval_size:16777216
2020-10-18 15:55:15.384 KST [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /home/byron1st/fabric-samples/test-network/configtx/configtx.yaml
2020-10-18 15:55:15.386 KST [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2020-10-18 15:55:15.386 KST [common.tools.configtxgen] doOutputBlock -> INFO 006 Writing genesis block
+ res=0
Creating network "net_test" with the default driver
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating peer0.org1.example.com ... done
Creating orderer.example.com    ... done
Creating peer0.org2.example.com ... done
CONTAINER ID        IMAGE                               COMMAND             CREATED             STATUS              PORTS                                   NAMES
4b65b0290455        hyperledger/fabric-peer:latest      "peer node start"   1 second ago        Up Less than a second   7051/tcp, 0.0.0.0:9051->9051/tcp        peer0.org2.example.com
852550ea91d6        hyperledger/fabric-orderer:latest   "orderer"           1 second ago        Up Less than a second   0.0.0.0:7050->7050/tcp                  orderer.example.com
023351d6bdee        hyperledger/fabric-peer:latest      "peer node start"   1 second ago        Up Less than a second   0.0.0.0:7051->7051/tcp                  peer0.org1.example.com
```

test-network 실행

- **test-network/docker/docker-compose-test-net.yaml 열기**
  - volumes 섹션에서 다음을 수정
    - ../orderer.example.com:/var/hyperledger/production/orderer
    - ../peer0.org1.example.com:/var/hyperledger/production
    - ../peer0.org2.example.com:/var/hyperledger/production
  - ./network.sh down 후 재시작

test-network 실행



하이퍼레저 패브릭 실행

- 채널 생성
- 체인코드 패키징 및 배포

채널 생성

- **채널**
  - Blockchain = Distributed Ledger
  - Channel = Ledger를 공유하는 Organization/Peer = Blockchain
    - Channel01: Org1의 Peer0, Org2의 Peer0가 가입
      - Org1의 Peer0와 Org2의 Peer0에만 Channel01의 Ledger가 저장됨
    - Channel02: Org1의 Peer0, Org3의 Peer0가 가입
      - Org1의 Peer0와 Org3의 Peer0에만 Channel01의 Ledger가 저장됨

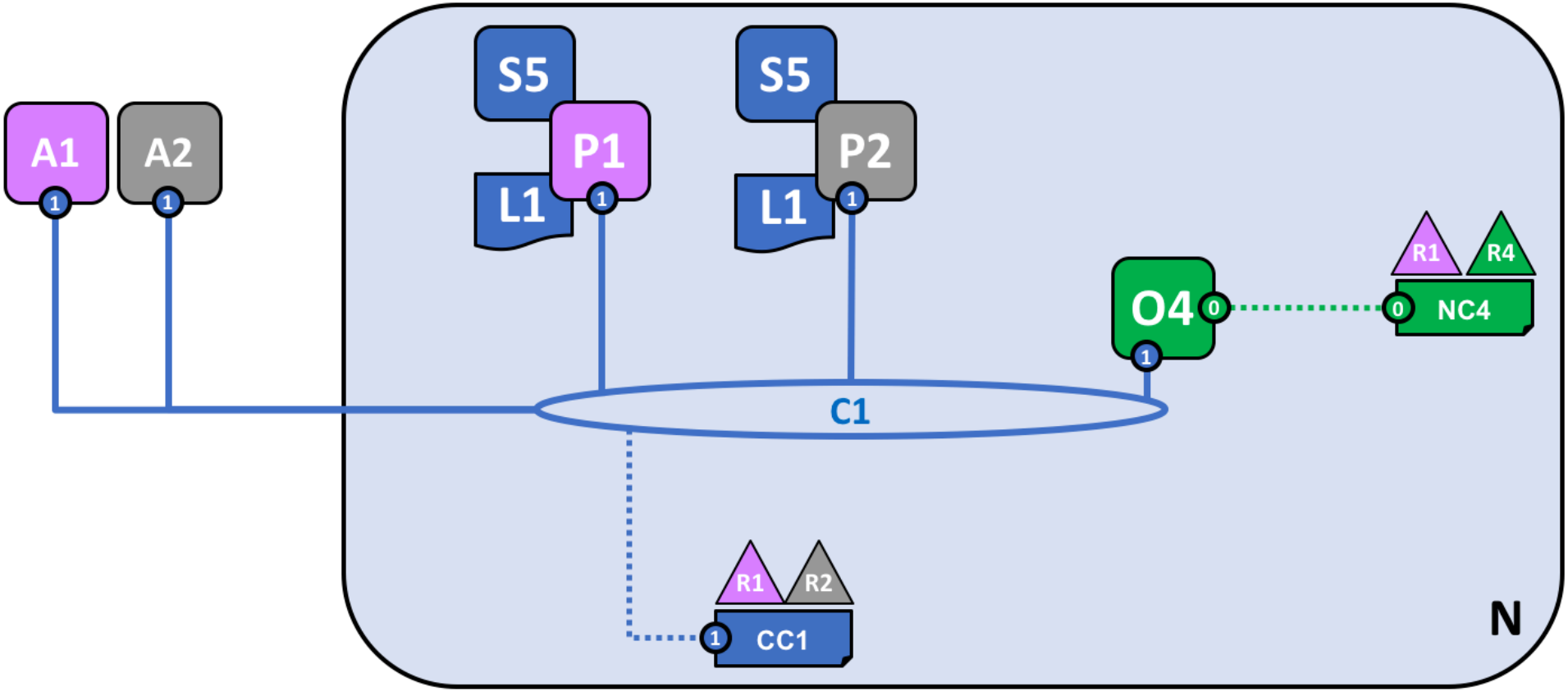
  - Channel 종류
    - System channel
      - Orderer가 처음 실행할 때 만드는 채널 → 사용자와는 크게 관련 없음
    - 그 외 Channel
      - 실질적으로 사용자의 트랜잭션들이 기록되는 곳

채널 생성

- 채널

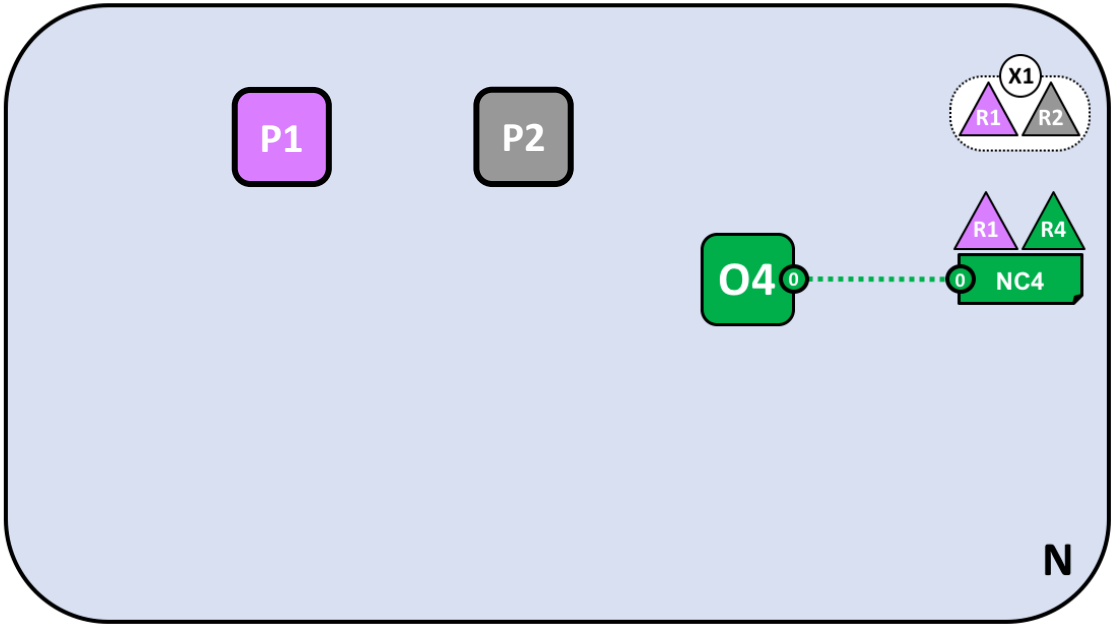
A: Client Server
S: Chaincode
P: Peer
O: Orderer
L: Ledger
C: Channel

CA: CA
R: Organization
CC: Channel Config
NC: Network Config



채널 생성

- **우리가 지금까지 한 것**
  - cryptogen 으로 orderer.example.com, org1.example.com, org2.example.com 기관의 msp, tls 키를 생성
  - configtx.yaml 정의 → Network Configuration
  - System channel을 위한 Genesis block 생성 → Orderer에 장착
  - Orderer 실행 (시스템 채널 생성됨)
  - Peer 실행 (아직 연결되지 않음)



채널 생성

- ./test-network 폴더에서 다음 커맨드 실행
  - ./network.sh createChannel

- chains/ 폴더 안에 채널별로 폴더가 생성되는 것 확인

체인코드 패키징 및 배포

- Prerequisite
- asset-transfer-basic 체인코드 배포 (JavaScript 버전)
  - ./network.sh deployCC -ccl javascript

```
byron1st@DESKTOP-D240NBD:~/fabric-samples/test-network$ docker ps
CONTAINER ID        IMAGE
     CREATED             STATUS              PORTS
320cffafe346        dev-peer0.org2.example.com-basic_1.0-3e6efb14d2337
…"   5 minutes ago       Up 5 minutes
1dad109710d8        dev-peer0.org1.example.com-basic_1.0-3e6efb14d2337
…"   5 minutes ago       Up 5 minutes
a74ab9620dc1        hyperledger/fabric-orderer:latest
     11 minutes ago      Up 11 minutes       0.0.0.0:7050->7050/tcp
7fab09212cb8        hyperledger/fabric-peer:latest
     11 minutes ago      Up 11 minutes       7051/tcp, 0.0.0.0:9051->9
c281d2758196        hyperledger/fabric-peer:latest
     11 minutes ago      Up 11 minutes       0.0.0.0:7051->7051/tcp
```

체인코드 패키징 및 배포

- Prerequisite
- asset-transfer-basic 체인코드 배포 (JavaScript 버전)
  - ./network.sh deployCC -ccl javascript
- 체인코드 패키징 및 배포 절차 (2.x 버전)
  - 체인코드 package – 체인코드 별로 수행
    - .tar.gz 형태로 메타 데이터와 함께 압축
      - mkdir basic_cc & & tar -xvf basic.tar.gz -C ./basic_cc
  - 체인코드 install – Peer 별로 수행
    - 바이너리 형태로 Peer에 저장
  - 체인코드 approve – 기관 별로 수행
    - version: 코드 기준
    - sequence: 설정 등을 변경해서 재배포 가능
  - 체인코드 commit – 채널 별로 수행

체인코드 패키징 및 배포

- **체인코드 함수 호출**
  - 로컬에 있는 peer 바이너리를 이용하여 체인코드 함수를 호출 해볼 것임.
  - 다음 커맨드들을 순서대로 호출

  - https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html#interacting-with-the-network

  - export FABRIC_CFG_PATH=$PWD/../config/
  - export CORE_PEER_TLS_ENABLED=true
  - export CORE_PEER_LOCALMSPID="Org1MSP"
  - export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
  - export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
  - export CORE_PEER_ADDRESS=localhost:7051

체인코드 패키징 및 배포

- **체인코드 함수 호출**
  - peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}'

  - docker logs 로그 확인

  - peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'

하이퍼레저 패브릭 Client Server 실행

- node.js 설치
- Client Server 실행

node.js 설치

- JavaScript
  - 프론트앤드, 백앤드에서 사용이 가능한 프로그래밍 언어
  - JavaScript 코드를 실행하기 위해서는 JavaScript 인터프리터 필요 -〉 JavaScript 엔진
    - Chrome (Google): V8
    - Firefox (Mozilla): SpiderMonkey
    - Safari (Apple): JavaScript Core
    - Edge(Microsoft): ~~Chakra Core~~ → V8

  - JavaScript 엔진을 웹 브라우저 밖에 설치하여 웹 브라우저 밖에서도 JavaScript 코드를 실행할 수 있도록 개발
    - V8: node.js(서버), Electron(데스크톱 어플리케이션)
    - JavaScript Core: React Native(iOS, Android 앱)

node.js 설치

- **node.js 설치**
  - Linux, macOS: Terminal 실행
  - Windows: Ubuntu 18.04 실행
  - 다음 커맨드 입력
    - curl -L https://raw.githubusercontent.com/tj/n/master/bin/n -o n
  - 다음 커맨드 입력
    - sudo bash n lts
  - 비밀번호 입력
  - 12.19.0 버전이 설치됨
  - 다음 커맨드 결과 버전 넘버가 잘 출력되는지 확인
    - node --version

Client Server 실행

- **네트워크를 CA 버전으로 다시 실행**
  - ./network.sh down
  - sudo rm -rf orderer.example.com/ peer0.org1.example.com/ peer0.org2.example.com/
  - ./network.sh up createChannel -c mychannel - ca
  - ./network.sh deployCC -ccn basic -ccl javascript

Client Server 실행

- **다음 폴더로 이동**
  - asset-transfer-basic/application-javascript
- **다음 커맨드 실행**
  - npm install

```
byron1st@DESKTOP-D240NBD:~/fabric-samples/asset-transfer-basic/application-javascript$ ls
app.js  node_modules  package-lock.json  package.json
```

- **서버를 실행**
  - node app.js
  - 다음 컨트랙들을 정해진 순서대로 실행하고 자동으로 종료됨
    - InitLedger
    - GetAllAssets
    - CreateAsset
    - ReadAsset
    - AssetExists
    - UpdateAsset
    - TransferAsset

Client Server 실행

- **Common Connection Profile**
  - Client Server가 필요로 하는 Channel, Peer, Orderer 정보가 담김
  - test-network/organizations/peerOrganizations/org1.example.com/connection-org1.json

- **사용자 Enroll**
  - "이 사용자를 이 PC에서 사용하겠습니다." 의미.
  - **End User** – 트랜잭션을 보내는 사람
    - CA에 register / enroll을 통해 Key 를 받아야 함 → **파일 또는 CouchDB**에 저장
  - **Admin User** – Fabric CA 관리자
    - Fabric CA를 사용할 경우, 사용자 register / enroll을 하려면 Fabric CA의 관리자 키가 필요함

Client Server 실행

- Fabric SDK
  - Gateway, Network, Contrac 객체
  - Gateway
    - Common Connection Profile
    - Identity (사용자 별로 객체 필요)
  - Network
    - Channel
  - Contract
    - submitTransaction
    - evaluateTransaction

```
await gateway.connect(ccp, {
    wallet,
    identity: org1UserId,
    discovery: { enabled: true, asLocalhost: true } //
});

// Build a network instance based on the channel where
const network = await gateway.getNetwork(channelName);

// Get the contract from the network.
const contract = network.getContract(chaincodeName);
```