하이퍼레저 패브릭 기반의 프라이빗 블록체인 개발 과정

2021-06-17

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

2021-06-17

빅픽처랩(주)

안휘

목차

1. 소프트웨어 서비스의 실제
2. 블록체인 기술
3. 블록체인 기술의 구성요소
4. 블록체인 기술이란

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 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)

2. 블록체인 기술

- 인간적 신뢰가 구축되지 않은 관계자들 5명이 존재

3. 블록체인 기술의 구성요소

- 공개키 암호화 기술

3. 블록체인 기술의 구성요소

- **디지털 인감도장: 공개키 암호화 기술 (PKI)**
  - 키: 암호화 또는 복호화를 하기 위해 사용되는 텍스트

3. 블록체인 기술의 구성요소

- **디지털 인감도장: 공개키 암호화 기술 (PKI)**

  - 암호화 기술
    - 단방향 암호화 기술
      - 복호화가 불가능

    - 비밀키 암호화 기술
      - 하나의 키로 암호화/복호화 모두 수행
      - 해당 키가 절대 유출되면 안되기 때문에 “비밀키” 암호화 기술

    - 공개키 암호화 기술
      - 고유한 2개의 키(암호화용 키, 복호화용 키)가 쌍으로 존재
      - 하나의 키로 암호화한 내용은 다른 키로만 복호화 가능

3. 블록체인 기술의 구성요소

- **디지털 인감도장: 공개키 암호화 기술 (PKI)**

  - PKI는 암호화와 복호화를 하기 위한 키를 2개 준비
    - 고유한 한 쌍의 키: 암호화용 키 1개, 복호화용 키 1개
    - 개인키로 암호화하면, 복호화는 같은 쌍의 공개키로만 가능

3. 블록체인 기술의 구성요소

- **디지털 인감도장: 공개키 암호화 기술 (PKI)**
  - PKI를 이용한 내용 검증

3. 블록체인 기술의 구성요소

- **디지털 인감도장: 공개키 암호화 기술 (PKI)**

  - 부동산 계약서의 **인감도장**
    - 나 = 인감도장
    - 증명 = 인감증명서 (정부가 보장)
  - 블록체인
    - 나 = 개인키
    - 증명 = 공개키를 통해 보장

3. 블록체인 기술의 구성요소

- 합의 기술

3. 블록체인 기술의 구성요소

3. 블록체인 기술의 구성요소

- **복사한 계약서 및 할인: 합의 기술 (Consensus)**

  - " 참여자 모두가 동일한 계약서를 가지고 있음 " 을 증명하는 기술
  - 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)**
  - Proof of Work: 노력에 의한 합의
  - Proof of Stake: 지분에 의한 합의
  - Fault Tolerance Algorithms: 리더에 의한 합의

3. 블록체인 기술의 구성요소

- **Proof of Work: 노력에 의한 합의**
  - 특정 조건에 맞는 숫자(Nonce, Number used once)를 찾는 작업
    - 빠르게 찾을 수 있는 방법 없음
    - 해당 조건에 맞는 숫자를 찾을 때까지 오로지 하나하나 해보는 방법 밖에 없음

    - 먼저 정답을 찾는 사람이 갖고 있는 계약서가 진품 (블록 생성 권한을 갖음)
      → 나머지 참여자들은 해당 내용을 복사해서 가지고 있음

3. 블록체인 기술의 구성요소

- Proof of Work: 노력에 의한 합의

- …근데 왜 이 짓(?)을 할까?
  - "블록생성권한"을 갖고, 블록을 만들면 정해진 양 만큼 보상이 주어짐
    → 보상을 "채굴"한다라고 표현
  - 네트워크가 유지되기 위해서는 누군가 문서를 검증하고, 확인하는 작업을 해주어야 함
  - 보상은 네트워크 유지에 대한 보상임
- 비트코인, 이더리움 등 대부분 이 방식을 따르고 있음

3. 블록체인 기술의 구성요소

- **Proof of Stake: 지분에 의한 합의**
  - 토큰 보유량에 따라 블록생성권한을 조정함
  - 구현에 따라 여러 모습이 존재
    - 토큰 보유량에 따라 블록생성 및 검증을 수행할 대표자를 선정
    - 토큰 예치 등을 통해 권한 부여
    - …
  - Proof of Work 의 과도한 에너지 사용 문제를 해결
  - 이더리움이 향후 이 방향으로 진화할 예정

3. 블록체인 기술의 구성요소

- **Fault Tolerance Algorithms: 리더에 의한 합의**

  - Fault Tolerance Algorithms
    - 참여자 중 일부가 불능상태(fault)가 되더라도, 네트워크를 유지시키는 알고리즘
    - 예:
      - 3대의 컴퓨터가 동일한 내용을 검증하여 저장 중
      - 1대의 컴퓨터가 갑자기 불능이 됨
      - 그래도 내용이 올바르게 검증되었다는 것을 보장함

    - Crash fault vs Byzantine fault
      - Crash fault: 참여자 중 일부가 불능 상태에 빠짐
      - Byzantine fault: 참여자 중 일부가 “배신자”가 됨

3. 블록체인 기술의 구성요소

- Fault Tolerance Algorithms: 리더에 의한 합의
  - Byzantine Fault

Coordinated Attack Leading to Victory　　Uncoordinated Attack Leading to Defeat

3. 블록체인 기술의 구성요소

- **Fault Tolerance Algorithms: 리더에 의한 합의**

  - Fault Tolerance Algorithms
    - 일반적으로 절반(Crash fault), 또는 1/3(Byzantine fault)의 참여자가 불능상태에 빠져도 합의가 올바르게 진행되는 것을 보장함
    - 소수의 참여자들이 빠르게 합의하기 때문에 다른 합의 대비 매우 빠르고, 에너지 소모가 거의 없음
    - 보통 참여자 숫자가 20~30 정도면 한계에 도달
    - 프라이빗 블록체인에서 주로 사용
    - 현재 기술은 Crash Fault 수준

3. 블록체인 기술의 구성요소

- 단방향 암호화 기술

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

3. 블록체인 기술의 구성요소

- 간인: 해시 기술 (Hash)

- 저장된 데이터를 위조하려면, 그 뒤에 저장한 모든 데이터를 위조해야 함
  -〉 위변조가 현실적으로 불가능

3. 블록체인 기술의 구성요소

- 간인: 해시 기술 (Hash)

3. 블록체인 기술의 구성요소

- **스마트 컨트랙트**
  - 블록체인 = 데이터 저장소

  - 스마트 컨트랙트 = 블록체인 입출력 프로그램
    - 블록체인에 데이터를 쓰고, 읽는 프로그램
    - 비트코인
      - 블록체인에 비트코인 전송 이력을 쓰고, 읽는 스마트 컨트랙트 1개만 존재
    - 이더리움
      - 개발자가 자유롭게 블록체인에 저장할 데이터를 결정하고, 이를 읽고 쓰는 프로그램을 만들 수 있음 = **스마트 컨트랙트 플랫폼**

3. 블록체인 기술의 구성요소

- **트랜잭션**

명사 뜻별예문

(격식)

1. [불가산] [업무·교섭·활동 등의] **처리,** 집행, 취급[of ··].

참고 [어원] trans-(이동)+action(행동)→행동으로 옮기다

2. (처리가 끝난 또는 처리 중인) **업무;** (종종 **transactions**) **(상)거래,** 매매

Internet [credit-card] transaction
인터넷 [신용카드]거래

- 스마트 컨트랙트 실행 내역
  - "A 스마트 컨트랙트의 A-1 기능을 "입력값"으로 실행하여, "출력값"을 저장했다"
  - "비트코인의 송금기능을 "from:철수, to:영희, amount: 100" 입력값으로 실행하여, "철수:-100, 영희:100"을 저장했다"

3. 블록체인 기술의 구성요소

- **트랜잭션, 블록, 분산원장**
  - 스마트 컨트랙트: 블록체인 입출력 프로그램
  - 트랜잭션: 스마트 컨트랙트 실행 내역
  - 블록: 검증된 트랜잭션들의 모음
  - 분산원장: 검증된 블록들의 모음

4. 블록체인 기술이란

- **블록체인이란 “데이터 저장” 기술**
- **비트코인 (암호화폐)**
  - 블록체인 기술을 이용해서 “**이체확인증**”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

4. 블록체인 기술이란

- 인감도장: 공개키 암호화 기술 공인인증서 기술
- 복사한 계약서 및 할인: 합의 기술 네트워크 기술
- 간인: 단방향 암호화 기술 비밀번호 저장 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 기술이란

- **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

4. 블록체인 기술이란

- **한계**
  - 프라이버시
    - 사실 블록체인과 관계없음.
    - 프라이버시는 서비스 설계에서 논할 일이지, 블록체인을 쓴다고 해결되는 문제가 아님!
    - 그래서 투명성이 장점인 블록체인에 개인정보를 올리는건 “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 공공의 목적을 달성할 경우, 성능 제약에도 도입 가능 =〉 블록체인이 공공 영역에서 더 논의되어야 하는 이유!

프라이빗 블록체인의 현재

2021-06-17

빅픽처랩(주)

안휘

목차

1. 블록체인 기술의 비전
2. 퍼블릭 블록체인 기술
3. 프라이빗 블록체인 기술
4. 국내 블록체인 기술

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. 퍼블릭 블록체인 기술

- **퍼블릭 vs 프라이빗**
  - 데이터저장소(블록체인)에 대한 접근 권한의 차이
  - 블록체인에 읽고 쓸 수 있는 사람은?
    - 누구나 가능: 퍼블릭 블록체인
    - 정해진 참여자만 가능: 프라이빗 블록체인

  - 퍼블릭 블록체인
    - 지금 당장 여러분도 컴퓨터를 키고 네트워크에 참여하여 블록을 읽고 쓸 수 있음
    - 비트코인, 이더리움, EOS 등

  - 프라이빗 블록체인
    - Permissioned 블록체인으로 불리우기도 함
    - 주로 기업들, 정부 등이 제한된 컨소시움을 만들어 구축

2. 퍼블릭 블록체인 기술

- **여러분이 익숙한 바로 그 블록체인들**
  - 비트코인, 이더리움, EOS 등

- **비트코인**
  - 2009년 1월 프로그램 배포 (블록체인 저장소를 읽고 쓸 수 있는 프로그램)
  - 오픈소스로 공개 후 개발: Bitcoin Core
    - 현재 0.21.1 버전
  - 기술적으로는 초기와 큰 차이가 없음
  - 블록체인에 거래 내역(이체확인증)을 저장

2. 퍼블릭 블록체인 기술

- **이더리움**
  - 화폐거래기록 뿐 아니라 여러 추가 정보를 기록할 수 있도록 함
    - 스마트 컨트랙트의 등장
    - 무엇을 저장할지 개발자가 정의할 수 있음
      → 수많은 어플리케이션 개발 가능성

  - 2013년 비탈릭 부테린이 백서를 작성하여 개발을 제안
  - 2014년 이더리움 재단 설립 및 개발 자금 펀딩
  - 2015년 7월 30일 비탈릭 부테린에 의해 개발됨
  - 2020 ~ 2021년: 이더리움 2.0 진화 예정

  - 여전히 끊임없이 발전 중
  - 가장 개발자 친화적 플랫폼
  - 대부분의 알트코인, 블록체인 어플리케이션(DeFi, NFT 등)은 이더리움의 스마트 컨트랙트로 구현됨

2. 퍼블릭 블록체인 기술

- **이더리움**
  - 문제점
    - 지나치게 비싼 이용료
      - 스마트 컨트랙트 구동을 위해 필요한 금액이 지나치게 비쌈
    - 너무 느린 실행 속도
      - Proof of Work 기반의 퍼블릭 블록체인 특성
      - 스마트 컨트랙트 실행이 확정되기까지 지나치게 긴 시간 소요
      - 평균 20 TPS (초당 처리되는 트랜잭션 수)

  - 이더리움 2.0
    - Proof of Stake 로 이전 (채굴로 인한 에너지 소모 감소)
    - 14,000 TPS 목표 (느린 속도 개선)
    - 확장성 개선 (더 많은 참여 유도)

3. 프라이빗 블록체인 기술

- **리플**
  - 전 세계 은행들의 실시간 국제 송금을 위한 암호화폐
  - 오래된 국제 송금 네트워크인 SWIFT 를 대체
  - 누구나 거래내역을 받아보고 트랜잭션을 발생시킬 수 있지만,
    트랜잭션 검증(블록 생성)과 네트워크 운영은 "리플사(社)"에 의해 운영됨

  - 2013년부터 BoA, HSBC 등이 참여하여 은행간 결제에 사용
  - 2021년 현재,
    - 40여개 은행 & 회사들이 참여 중
    - 50여개 은행 & 회사들이 실험적으로 참여 중

출처: "Banks That Are Currently Using Ripple XRP Technology"
(https://www.coinsuggest.com/bank-ripple-xrp/), 2021

3. 프라이빗 블록체인 기술

- **리플**
  - SWIFT 망을 통한 송금
  - SWIFT: 세계은행간 금융데이터통신협회
    - 국제간 대금결제 등을 위한 데이터 통신 연결망을 기획 & 운영
    - 1973년 설립된 비영리기관

이미지: Building Network Effects on Ripple, 2015

3. 프라이빗 블록체인 기술

- **리플**
  - 리플넷이 가운데서 송금을 중계

이미지: Building Network Effects on Ripple, 2015

3. 프라이빗 블록체인 기술

- Hyperledger
  - 기업용 블록체인 기술을 연구, 개발하는 오픈소스 커뮤니티
  - 운영: Linux Foundations
  - 참여:
    - IBM, 인텔, JP 모건, 후지쯔, 히타치, 바이두, 화웨이, 보쉬, 다임러, LG CNS, 삼성 SDS, 월마트 등 174개 기업/조직/학교

HYPERLEDGER

이미지: Hyperledger Architecture, Vol II.

3. 프라이빗 블록체인 기술

- Hyperledger
  - Hyperledger Fabric
    - 프라이빗 블록체인 소프트웨어
    - 네트워크 구성, 스마트 컨트랙트 개발 등이 가능
    - 블록체인기반 소프트웨어 서비스 개발의 기반이 되고 있음

HYPERLEDGER

4. 국내 블록체인 기술

- **클레이튼 (Klaytn) – 그라운드X (카카오)**
  - 퍼블릭 블록체인 기반으로 프라이빗 블록체인 통합
  - 이더리움과 유사한 모습을 추구

4. 국내 블록체인 기술

- **넥스레저(Nexledger) – 삼성SDS**
  - 기업용 블록체인 솔루션
  - 단순 블록체인 저장소 외에 각종 기업용 소프트웨어 서비스(관리, 운영 등)를 추가
  - 3가지 버전 존재
    - 독자 블록체인 기술, Hyperledger Fabric, 이더리움
  - 실제/시범 운영사례
    - 카드사 문서관리, 계정 로그인 서비스, 은행권 고객인증, 지자체 스마트투표 등
    - 운송/물류에 시범사업

- **모나체인(Monachain) – LG CNS**

Monachain
LG CNS Blockchain Platform

  - 기업용 블록체인 솔루션
  - 단순 블록체인 저장소 외에 각종 기업용 소프트웨어 서비스(관리, 운영 등)를 추가
  - 한국조폐공사의 블록체인 지역화폐 서비스 Chak에 적용
    - 투표, 지역/캠퍼스 화폐, 인증

하이퍼레저 패브릭을 구성하는 기술들

2021-06-17

빅픽처랩 ㈜

안 휘

목차

- 하이퍼레저
- 하이퍼레저 패브릭

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 (LTS): 2019.01
  - 2.0: 2020.01
  - **2.2 (LTS): 2020.7**
  - 2.3: 2020.11
  - 2.4 (alpha): 2021.4

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

2021-06-17

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” → “시스템” → “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 18.04” 검색
    - “Ubuntu 18.04” 설치
      - Microsoft 로그인 필요 없음
    - 다운로드 완료 후 실행
    - 사용자 계정 생성
      - 비밀번호 입력 때는 원래 아무 텍스트도 타이핑 되지 않으니 당황하지 말 것

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)
  - 설치 완료 후 Close And Logout 을 클릭하여 로그아웃 했다가 다시 접속

docker, docker-compose 설치

```
byron1st@HWIAHN7D0D:~$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2F
var%2Frun%2Fdocker.sock/v1.24/containers/json: dial unix /var/run/docker.sock: connect: permission denied
byron1st@HWIAHN7D0D:~$ sudo usermod -aG docker $USER
[sudo] password for byron1st:
byron1st@HWIAHN7D0D:~$
```

- Get permission denied 에러가 발생할 경우
  - **`sudo usermod -aG docker $USER`**
  - 비밀번호 입력
  - Ubuntu 껐다가 다시 실행

하이퍼레저 패브릭 설치

- 하이퍼레저 패브릭 설치
- test-network 실행

하이퍼레저 패브릭 설치

- Docker 를 실행
- Ubuntu 를 실행
- 다음 커맨드 입력
  - `curl -sSL https://bit.ly/2ysbOFE | bash -s -- 2.2.3 1.4.9`
- 명령어를 실행한 곳으로부터 “./fabric-samples” 폴더에 복사됨
  - 버전은 최신 버전인 2.2.3이 설치됨

test-network 실행

- Ubuntu 를 실행
- 다음 커맨드 입력
  - `cd ~/fabric-samples/test-network`
- 다음 커맨드 입력
  - `export COMPOSE_PROJECT_NAME=fabric`
  - `./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

- ~/fabric-samples 위치에서 Visual Studio Code 실행
  - `code .`

- test-network/docker/docker-compose-test-net.yaml 열기
  - volumes 섹션에서 다음을 수정
    - ./blocks/orderer.example.com:/var/hyperledger/production/orderer
    - ./blocks/peer0.org1.example.com:/var/hyperledger/production
    - ./blocks/peer0.org2.example.com:/var/hyperledger/production
  - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=test
  - ./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 (2.3 버전 이후 삭제)
      - 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 -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript -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)

- 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 PATH=${PWD}/../bin:$PATH`**
  - **`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 install 12`**
  - 비밀번호 입력
  - 12.22.1 버전이 설치됨
  - 다음 커맨드 결과 버전 넘버가 잘 출력되는지 확인
    - **`node --version`**

Client Server 실행

- **네트워크를 CA 버전으로 다시 실행**
  - (fabric-samples/test-network) 에서 수행
  - **./network.sh down && sudo rm –rf ./docker/blocks**
  - **./network.sh up -ca**
  - **./network.sh createChannel**
  - **./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript -ccl javascript**

Client Server 실행

- 다음 폴더로 이동
  - `cd ../asset-transfer-basic/application-javascript`
- 다음 커맨드 실행
  - `sudo apt-get update`
  - `sudo apt-get install build-essential`
  - `npm install`

- 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, Contract 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);
```