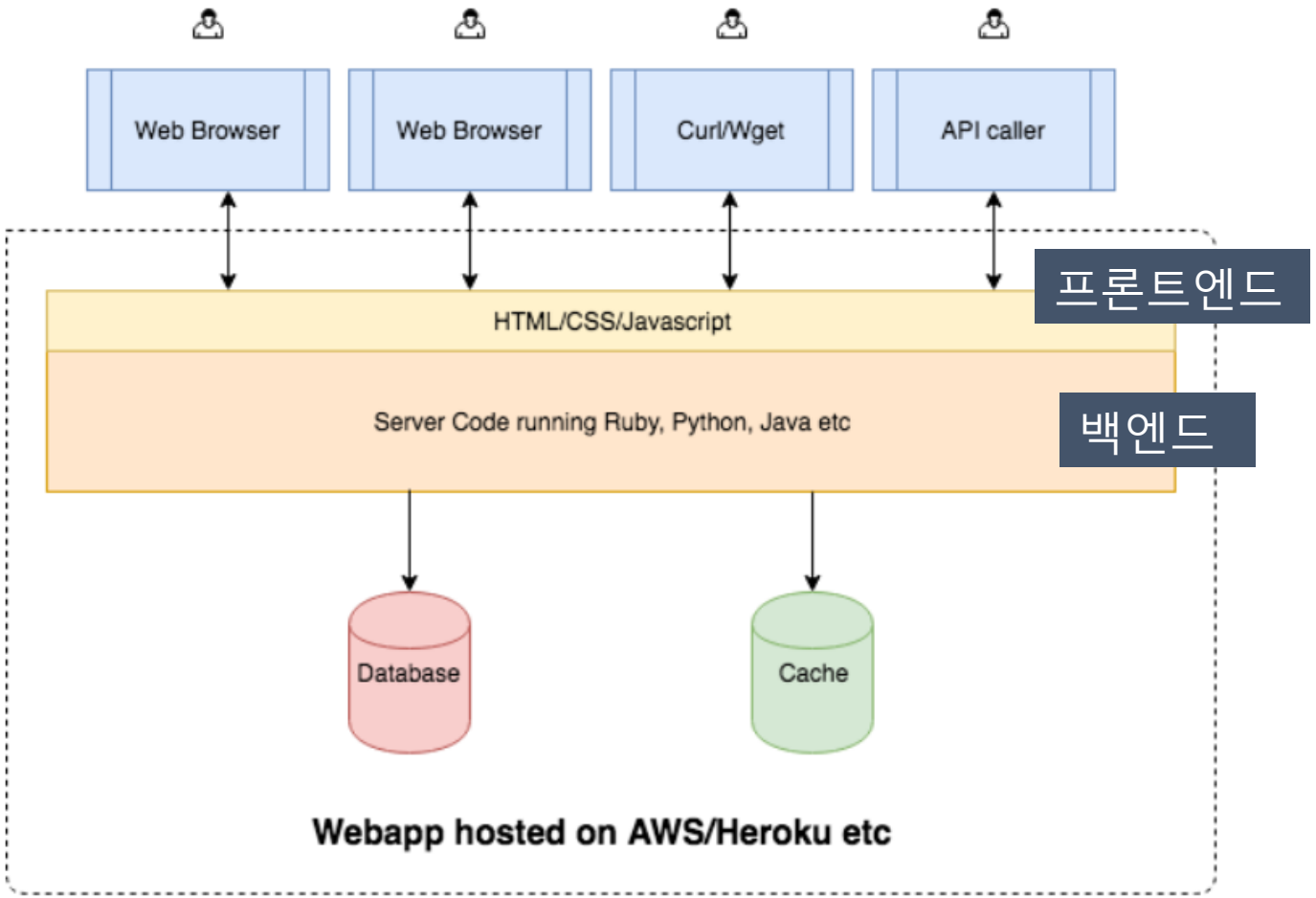
Block Chain

이더리움과 솔리디티

한경대학교 조한결

목차

01 이더리움 아키텍처와 사설망
기존 아키텍처와 이더리움 아키텍처

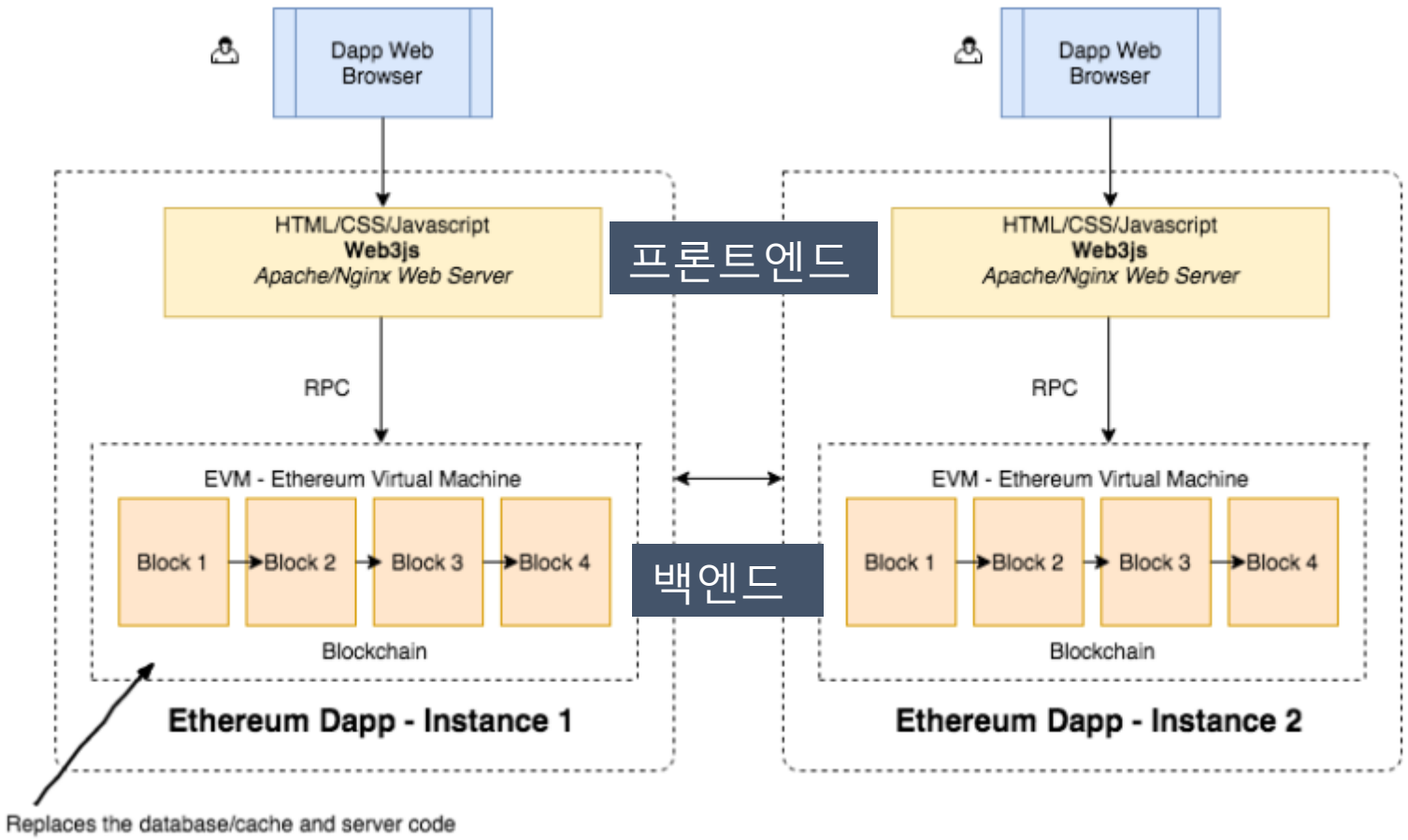


출처 : https://kr.zastrin.com/courses/4/lessons/2-2

기존의 클라이언트/서버 아키텍처

- 사용자는 웹브라우저나 API를 통해 중앙화 된 하나의 웹 애플리케이션과 상호작용한다.
- 클라이언트가 서버에 요청하면 서버는 요청을 처리한다. 그리고 DB 또는 캐시와 통신해 읽고, 쓰고, 업데이트 후 클라이언트에 응답을 반환

01 이더리움 아키텍처와 사설망
기존 아키텍처와 이더리움 아키텍처



출처 : https://kr.zastrin.com/courses/4/lessons/2-3

이더리움 클라이언트/서버 아키텍처

- 블록체인의 역할

1. 데이터베이스 : 모든 트랜젝션은 블록체인에 저장된다. (공개, 조작불가)
2. 코드 : 솔리디티로 만들어진 코드(계약서)를 컴파일 후 블록체인에 저장, 실행한다.

- 기본적으로 블록체인은 데이터 및 코드를 저장하며, 코드를 EVM(이더리움 가상머신)에서 실행한다.

01 이더리움 아키텍처와 사설망
이더리움

비트코인과 이더리움의 가장 결정적 차이

비트코인의 스크립트 언어는 비교적 단순해서 비트코인이 '화폐'로서만 작동하게 한다.

⇨ ‘튜링 불완전성(Turing-incompleteness)’

ethereum

이더리움의 개발자 비탈릭은 비트코인의 스크립트보다 더 세련된 언어(튜링 완전한 언어)를 구사하려고 새로운 블록체인 네트워크를 만들어 **여러가지 dApp(분산 어플리케이션)을 이용할 수 있는 플랫폼**을 만들기로 하는데 이것이 이더리움 플랫폼이다.

⇨ ‘튜링 완전성(Turing-completeness)’

01 이더리움 아키텍처와 사설망
네트워크 유형

메 인 넷

여러 사람이 실제로
이더리움을 사용하는
네트워크

테 스 트 넷

롭튼(Ropten)

코밴(Kovan)

린키비(Rinkeby)

개발에 사용하는
메인넷과 같은 구조의
임시 네트워크

프 라 이 빗 넷

개발 중 필요에 따라
개발자가 만드는
네트워크

테스트 순서

01 이더리움 아키텍처와 사설망
Go Ethereum : Geth 클라이언트

Geth는 이더리움 재단(Ethereum Foundation)이 제공하는 공식 클라이언트 소프트웨어로써, Go언어로 개발되었다.

Geth는 블록체인의 복사본을 최신 상태로 유지하기 위해 끊임없이 다른 노드와 통신한다. 또한 블록을 채굴하고, 블록체인에 트랜잭션을 추가하고 블록의 트랜잭션을 검증하며 트랜잭션을 실행할 수도 있다.

그 외에 사용 가능한 클라이언트..

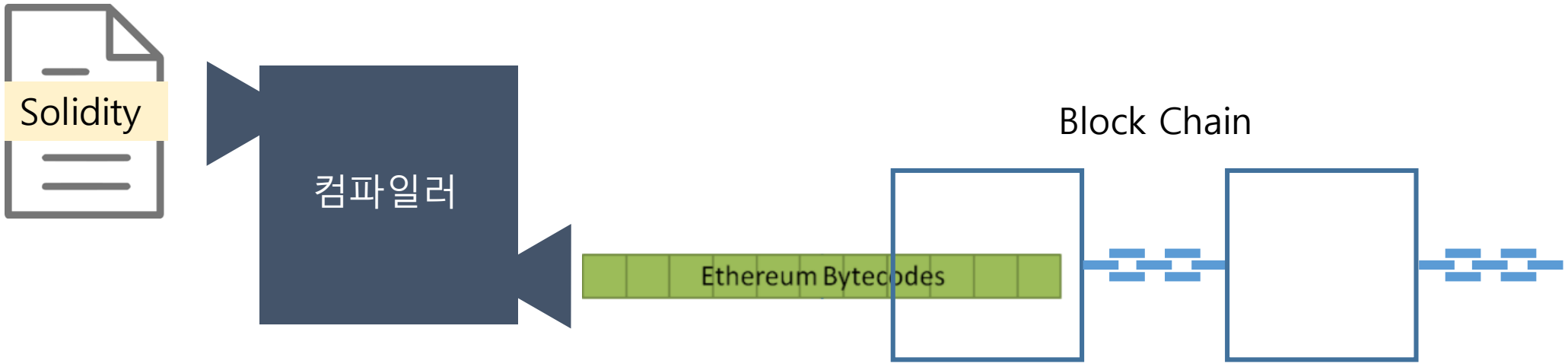
가나슈(Ganache)

- 메모리 내 블록체인을 사용하는 방법
- 트랜잭션의 실행 시간 단축
- npm install ganache-cli

솔리디티(Solidity)

스마트 계약을 구현하기 위한 객체지향언어. 이더리움 상태에서 계정의 동작을 제어하기위한 언어이다.
Ethereum Virtual Machine (EVM)을 목표로 설계되었다.

문서 : https://solidity.readthedocs.io/en/develop/index.html



목차

컨트럭트

컨트랙트는 이더리움 애플리케이션의 기본적인 구성 요소로, 모든 변수와 함수는 어느 한 컨트랙트에 속하게 된다. 다른 객체지향언어의 클래스(Class) 역할을 한다.

```
1   pragma solidity ^0.4.19;// 여기에 솔리디티 버전 적기
2
3   contract HelloWorld { //  컨트랙트 생성
4
5   }
```

상태 변수와 정수

상태 변수는 컨트랙트 저장소에 영구적으로 저장된다. => 이더리움 블록체인에 기록

솔리디티의 정수 타입은 부호가 있거나 없는 정수를 지정 할 수 있다.

```
1   pragma solidity ^0.4.19;
2
3   contract Integers {
4       // 부호없는 정수
5       uint8 a = 3;
6       uint b = 2; // = uint256
7       // 부호있는 정수
8       int c = 4;
9       int d = -5;
10  }
```

왼쪽 값을 포함하는 연산자

```
1   pragma solidity ^0.4.19;
2
3   contract Operator{
4       // a = a + e
5       a += e;
6       // a = a - e
7       a -= e;
8       // a = a * e
9       a *= e;
10      // a = a / e
11      a /= e;
12      // a = a % e
13      a %= e;
15      // a = a | e
16      a |= e;
17      // a = a & e
18      a &= e;
19      // a = a ^ e
20      a ^= e;
21      // 증가 및 감소 연산자
22      a++;
23      a--;
24  }
```

02 솔리디티 문법과 기본
구조체와 배열

구조체

```
struct Person {
    uint age;
    string name;
}
```

배열

```
18  contract Arrays {
19      // 고정 배열
20      uint[5] fArray = [uint(1), 2, 3, 4, 5];
21
22      // 동적 배열
23      uint[] dArray;
24  }
```

함수 선언

```
function eatFood(string _food, uint _cal) {
    // 함수의 내용
}
```

Public / Private

함수명 다음에 선언, 기본적으로 public 으로 선언

```
function _addToArray(uint _number) private {
    numbers.push(_number);
}
```

리턴값

```
function sayHello() public returns (string) {
    return greeting;
}
```

리턴값 필요시 리턴 할 값 정의

Reference

- https://kr.zastrin.com/courses/kr-simple-voting/lessons/2-2
- https://programmers.co.kr/learn/courses/7322/lessons/42394
- http://hellogohn.com/post_one259
- https://steemit.com/kr/@jsralph/4r1deg-1