

* DApp

1) 일부 리소스를 중앙화된 서버에 저장 (ex 크립토키티)

2) 100% 블록체인 위에서 서비스 (ex DelegateCall)

* 프론트엔드

1) HTML/CSS/Javascript : 웹 브라우저 기반으로 만들 때 웹 프론트엔드 작성에 사용

2) Web3.js : 이더리움 자바스크립트 API

3) Metamask : 크롬 및 파이어폭스 확장 프로그램, 일부 필수적인 블록 헤더 데이터만 외부의 풀노드로부터 받아와 검증함. 여러 사이트의 계정을 관리하고 블록체인 트랜잭션에 서명할 수 있음

* 백엔드

1) Solidity : 이더리움의 스마트 컨트랙트를 작성할 수 있는 객체지향형 프로그래밍 언어

2) Remix IDE : 솔리디티 코드를 작성하고 컴파일 및 배포를 돕는 온라인 컴파일러 툴 (사용 시 solc 설치 안해도됨)

3) TestRPC(Ganache) : 개발 단계에서 실제 이더없이 시뮬레이션 테스트 환경을 구성할 수 있게 하는 툴. 마이닝 없이 가상환경에서 리소스,트랜잭션에 대한 제한 없이 이더리움을 만들어내서 테스트 할 수 있음

4) TestNet : 이더리움의 퍼블릭 테스트 네트워크

현재 제공되는 퍼블릭 테스트 네트워크

1. Ropsten (Proof of Work) : Geth와 Parity 클라이언트 둘 다 지원, 실제 메인넷과 가장 비슷함

이더 마이닝 O, 테스트 이더 요청 O

1. Kovan (Proof of Authority) : Parity 클라이언트만 지원, 스팸 어택으로부터 안전

이더 마이닝 X, 테스트 이더 요청 필요

1. Rinkeby (Proof of Authority) : Geth 클라이언트만 지원, 스팸 어택으로부터 안전,

이더 마이닝 X, 테스트 이더 요청 O

5) MainNet : 실제 거래가 이루어지는 이더리움의 메인 네트워크

* DApp 개발 절차

1) Solidity로 스마트 컨트랙트 만들기

- Remix IDE 사용

2) 테스트넷 배포

- 이더리움 블록체인에 한 번 배포한 컨트렉트는 지울 수 없음

- 컨트렉트 배포 절차

1. TestRPC(Ganache)와 같은 가상 환경에서 솔리디티 컨트랙트 테스트 (실제 이더 X)
2. 테스트넷에 배포 (실제 ㄴ블록체인, 채굴을 통해 이더를 생성해야 하지만 채굴 난이도가 낮음)
3. 이더리움 메인넷에 배포 (모든 트랜잭션 발생할 때마다 가스를 소모 = 실제 이더 지불)

3) Front-end 만들기

- html 파일을 만듦

- Web3.js : 컨트렉트와 데이터를 주고 받을 수 있도록 정해진 규약대로 메시지를 만들어줌 (웹 클라이언트를 만들 때 써야 함)

- ABI(Application Binary Interface) : 언어나 플랫폼에 대한 의존성 없이 어디서나 해당 인터페이스에 대한 정보를 제공할 수 있게함 (함수에 대한 설명, payable, stateMutability 등 포함)

4) 메인넷 배포