Truffle & Ganache 사용법

truffle을 사용하여 DApp 개발환경 구성을 하는 방법을 알려드리는 튜토리얼입니다.
초보자들도 쉽게 따라할 수 있게 만들었습니다.

기본적인 코딩은 할 줄 아는 분들을 대상으로 작성되었습니다. 물론 코딩에 익숙하지 않은 분들도 따라할 수 있게 최대한 자세하게 설명하였습니다. 하지만 독자들이 갖고 있는 사전 지식의 정도에 따라서 이해하시는 정도도 달라질 것입니다.

또한 터미널, cmd 혹은 Powershell을 다루는 법에 대해서는 기본적으로 알고 있다고 가정하고 글을 작성하였습니다.

먼저 Truffle이란?

> Truffle은 스마트 컨트랙트 개발, 컴파일, 배포 그리고 테스팅을 쉽게 할 수 있도록 도와주는 프레임워크입니다.

윈도우 환경에서 cmd 혹은 powershell을 실행할때는 꼭 관리자 권한으로 실행을 해주세요.

Truffle 설치법 - 사전 설치

Truffle을 설치하기 위해서는 사전에 설치되어야 하는 것이 있습니다.
크게 4가지입니다.
Node.js, Truffle, VS-Code(혹은 기타 IDE), 그리고 Ganache 입니다.

하나씩 간단하게 설명드리도록 하겠습니다.

Node.js 설치

아래에서 본인 해당되는 OS를 선택하고 그에 맞게 설치하면 됩니다.

▼ **Windows**

1. 설치 링크인 https://nodejs.org/ko/ 에 들어갑니다.

2. 되도록이면 안정적, 신뢰도 높음이라고 적혀있는 버전을 다운로드 받고 node.js를 설치합니다.
3. window + R을 누른 후에, cmd라고 치고 실행시킵니다.
   혹은 powershell을 검색하여 powershell을 켭니다.
4. node.js의 버전을 확인하는 명령어를 치고 결과가 나오면 잘 설치된 것입니다.

```
(아래의 명령어를 칩니다.)
node -v

// 위의 명령어를 쳤을 때, 결과가 아래와 같이 v와 함께 특정 숫자가 나와야
v13.7.0 // (예시입니다. 상황에 따라서 다를 수 있습니다. )

npm -v
7.12.0 // (예시입니다. 상황에 따라서 다를 수 있습니다. )
```

▼ **MacOS**

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/

brew install node

brew update
```

```
(아래의 명령어를 칩니다.)
node -v

// 위의 명령어를 쳤을 때, 결과가 아래와 같이 v와 함께 특정 숫자가 나와야
v13.7.0 // (예시입니다. 상황에 따라서 다를 수 있습니다. )

npm -v
7.12.0 // (예시입니다. 상황에 따라서 다를 수 있습니다. )
```

```
// 업그레이드가 요구되면
brew upgrade node brew upgrade yarn
```

Truffle 설치

아래에서 본인이 해당되는 OS를 선택하고 그에 맞게 설치하면 됩니다.

▼ Windows

1. 앞에서 설치한 node.js를 이용하여 명령어로도 설치할 수 있다.

   아래의 명령어를 사용하면 된다.

   ```
   npm install -g truffle
   ```

2. 설치가 된 후에는 아래의 명령어로 확인할 수 있다.

   ```
   truffle version

   //아래와 비슷한 형식이 나오면 됨

   /* Truffle v5.1.56 (core: 5.1.56)
   Solidity v0.5.16 (solc-js)
   Node v13.7.0
   Web3.js v1.2.9 */
   ```

   만약에 저 위의 결과물들이 나타나지 않으면, 설치가 제대로 되지 않은 것이다.

▼ MacOS

1. 앞에서 설치한 node.js를 이용하여 명령어로도 설치할 수 있다.

   아래의 명령어를 사용하면 된다.

   ```
   npm install -g truffle
   ```

2. 설치가 된 후에는 아래의 명령어로 확인할 수 있다.

```
truffle version

//아래와 비슷한 형식이 나오면 됨

/* Truffle v5.1.56 (core: 5.1.56)
Solidity v0.5.16 (solc-js)
Node v13.7.0
Web3.js v1.2.9 */
```

만약에 저 위의 결과물들이 나타나지 않으면, 설치가 제대로 되지 않은 것이다.

Ganache 설치

아래에서 본인이 해당되는 OS를 선택하고 그에 맞게 설치하면 됩니다. 아래는 ganache-cli를 설치하는 방법이 소개되어있다. 조금 더 시각적으로 편한 상태에서 실습을 원한다면 ganahce 자체를 다운 받는 것을 추천한다. (아래로 이동)

▼ **Windows**

ganache 역시 2가지 방법으로 설치할 수 있다. 첫번째는 web에서 다운받아서 설치할 수 있다. 이 주소로 가면 현재 해당하는 OS에 맞는 버전으로 설치할 수 있다.

또는 아래의 명령어를 사용하여 설치할 수 있다.

```
npm install -g ganache-cli

ganache-cli --version
// Ganache CLI v6.12.2 (ganache-core: 2.13.2)
```

일단 해당 튜토리얼에서는 web에서 설치하고 진행할 계획이다.

▼ **MacOS**

ganache 역시 2가지 방법으로 설치할 수 있다. 첫번째는 web에서 다운받아서 설치할 수 있다. 이 주소로 가면 현재 해당하는 OS에 맞는 버전으로 설치할 수 있다.

또는 아래의 명령어를 사용하여 설치할 수 있다.

```
npm install -g ganache-cli

ganache-cli --version
// Ganache CLI v6.12.2 (ganache-core: 2.13.2)
```

일단 해당 튜토리얼에서는 web에서 설치하고 진행할 계획이다.

ganache는 링크에 들어가서 직접 다운로드를 받을 수 있다.

자 이제 사전설치는 모두 마무리가 되었습니다. 본격적으로 truffle 사용법에 대해서 알아봅시다.

Truffle 사용

▼ Truffle 사용

☐ **윈도우 환경에서 cmd 혹은 powershell을 실행할때는 꼭 관리자 권한으로 실행을 해주세요.**

truffle에는 다양한 종류의 명령어가 있습니다. 그 명령어를 모두 알 필요는 없지만, 이번 튜토리얼에서 사용할 것들은 미리 알려드리려고 합니다. 해당 튜토리얼에서 사용할 명령어는 2가지입니다. compile과 migrate이다.

다른 명령어는 어떤 것들이 있는지 알아보고 싶다면, 터미널에 truffle이라고 치면 아래와 같이 사용할 수 있는 명령어들의 종류가 나오게 됩니다.

```
// 터미널
truffle
```

```
                    MacBook-Pro ~ % truffle
Truffle v5.1.56 - a development framework for Ethereum

Usage: truffle <command> [options]

Commands:
  build     Execute build pipeline (if configuration present)
  compile   Compile contract source files
  config    Set user-level configuration options
  console   Run a console with contract abstractions and commands available
  create    Helper to create new contracts, migrations and tests
  debug     Interactively debug any transaction on the blockchain
  deploy    (alias for migrate)
  develop   Open a console with a local development blockchain
  exec      Execute a JS module within this Truffle environment
  help      List all commands or provide information about a specific command
  init      Initialize new and empty Ethereum project
  install   Install a package from the Ethereum Package Registry
  migrate   Run migrations to deploy contracts
  networks  Show addresses for deployed contracts on each network
  obtain    Fetch and cache a specified compiler
  opcode    Print the compiled opcodes for a given contract
  publish   Publish a package to the Ethereum Package Registry
  run       Run a third-party command
  test      Run JavaScript and Solidity tests
  unbox     Download a Truffle Box, a pre-built Truffle project
  version   Show version number and exit
  watch     Watch filesystem for changes and rebuild the project automatically

See more at http://trufflesuite.com/docs
```

보시는 것처럼, compile과 migrate이 포함되어 있는 것을 알 수 있습니다.

compile은 smart contract 코드를 compile시키는 명령어이며, deploy는 contract를 블록체인 네트워크에 배포할 때 사용되는 것이다.

local이 아닌 testnet 혹은 메인넷에 배포할 때는 network 명령어가 같이 사용된다. (추후에 사용할 예정이다.)

시작

truffle 프로젝트의 시작은 3단계로 이루어진다. **1) 특정 위치에 폴더 생성, 2) truffle init 3) 폴더 구조 확인** 이다.

1) 폴더 생성

폴더 생성은 평소와 같이 진행하면 됩니다. 혹은 아래의 명령어를 이용해도 된다.

```
// truffle_start라는 폴더를 생성
mkdir truffle_start

// truffle_start라는 폴더로 이동
cd truffle_start
```

2) truffle init

생성한 폴더에 이동한 후에 truffle 프로젝트를 시작하는 명령어를 실행시킨다.

```
// truffle_start 폴더에서
truffle init
```

위의 명령어를 실행시킨 후에는 아래와 같이 폴더 구성이 변화하게 된다.

```
ls // 하위에 어떠한 폴더들이 있는지 확인 (cmd는 dir)

// truffle_start 안에
|
|___ contracts
|
|___ migrations
|
|___ test
|
|___ truffle-config.js
```

3) 스마트 컨트랙트 확인

이제 스마트 컨트랙트를 확인해보자. 스마트 컨트랙트 파일들은 contracts 폴더 아래에 있다.

```
//truffle_start
truffle create contract A
cd contracts // contracts 폴더로 이동
```

contracts 폴더로 이동하면 아래와 같이 solidity 파일을 확인할 수 있다.

```
// truffle_start/contracts
|___ A.sol
```

추후에 solidity 파일을 추가할 때 이 위치에 추가하면 된다.

4-1) 간단하게 코드 입력

간단하게 코드를 입력해보겠다.

```
// truffle_start/contracts
cat A.sol // A.sol에 뭐가 있는지 확인

// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;
contract A {
    constructor() public{

    }
}

// vim A.sol // A.sol 안에 들어가기
```

```
// truffle_start/contracts/A.sol

i // insert 하기

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract ADD {
```

```
  function add(uint a, uint b) public pure returns(uint) {
    return a+b;
  }
}

// :wq

cat A.sol // A.sol에 뭐가 있는지 확인
```

💡 컴파일을 매우 간단히 설명하자면, 내가 짠 코드를 기계가 읽을 수 있게 번역하는 과정을 의미한다.

4-2) compile하기

작성한 스마트 컨트랙트를 이제 compile 할 것이다. compile은 아래의 명령어를 이용하여 실행한다.

```
truffle compile
```

오류가 일어나지 않으면, **Compiled successfully**이라는 아래의 메세지를 볼 수 있다.

```
Compiling your contracts...
===========================
> Compiling .\contracts\A.sol
> Compiling .\contracts\A.sol
> Artifacts written to C:\Users\garfi\OneDrive\Desktop\truff
> Compiled successfully using:
    - solc: 0.8.17+commit.8df45f5f.Emscripten.clang
```

💡 컴파일을 매우 간단히 설명하자면, 내가 짠 코드를 기계가 읽을 수 있게 번역하는 과정을 의미한다.

4-3) ganache-cli

다른 powershell 창을 만든 후에, ganache-cli 명령어를 입력하고 실행한다.

5-1) migrations js 파일 생성

truffle-config.js를 살펴보면 거의 모든 부분이 주석 처리가 되어있다. 약 67번째 줄부터 살펴보면(운영체제 및 버전마다 다를 수 있음) 아래와 같은 코드를 확인할 수 있다. 아래의 총 5줄의 주석을 해제하자. (CTRL+K+U)

```
// 장소 : migrations 폴더 안 1_add.js
const 변수명 = artifacts.require("solidity contract명" /*여기서

module.exports = function (deployer) {
  deployer.deploy(변수명);
};
```

5-2) truffle-config.js 수정 (migrate만)

truffle-config.js를 살펴보면 거의 모든 부분이 주석 처리가 되어있다. 약 67번째 줄부터 살펴보면(운영체제 및 버전마다 다를 수 있음) 아래와 같은 코드를 확인할 수 있다. 아래의 총 5줄의 주석을 해제하자. (CTRL+K+U)

```
// development: {
    //  host: "127.0.0.1",     // Localhost (default: none)
    //  port: 8545,            // Standard Ethereum port (de
    //  network_id: "*",       // Any network (default: none
    // },
```

또 1개의 더 주석을 해제해야한다. 약 109번째 줄을 살펴보면(운영체제 및 버전마다 다를 수 있음) 버전을 0.8.19로 수정한다.

```
compilers: {
    solc: {
      version: "0.8.19",    // Fetch exact version from solc
      // docker: true,        // Use "0.5.1" you've installe
      // settings: {          // See the solidity docs for a
      //  optimizer: {
      //    enabled: false,
      //    runs: 200
```

```
      //  },
      //  evmVersion: "byzantium"
      // }
    }
  },
```

version 수정, network 수정은 추후에, 구성설명

6) migrate하기

```
truffle migrate
```

성공을 하면 아래와 같은 메세지를 반환받게 된다.

```
Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'development'
> Network id:      1620362480211
> Block gas limit: 6721975 (0x6691b7)


1_add.js
========

   Deploying 'ADD'
   ---------------
   > transaction hash:    0x15f7d49b8a0ff3566480028d6b19bc43
   > Blocks: 0            Seconds: 0
   > contract address:    0x901503e3e5cE0bA613df8D9a83Ca1daaA
   > block number:        1
```

```
   > block timestamp:     1686187595
   > account:             0x4136F3ac062967CA3FE865Ffce8E32bC
   > balance:             99.999504215875
   > gas used:            146899 (0x23dd3)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.000495784125 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:      0.000495784125 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.000495784125 ETH
```

7) ganache 확인

동시에 ganahce-cli에서는 아래와 같은 메세지를 볼 수 있을 것이다.

```
eth_blockNumber
net_version
eth_accounts
eth_getBlockByNumber
eth_accounts
net_version
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_getBlockByNumber
eth_estimateGas
net_version
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction
```

```
  Transaction: 0x1824d81c9d195a38f076e41e301f4b61d073bbef03f9
  Contract created: 0x3287eb1fbed70373b9fec0227f041d0b8df994e
  Gas usage: 191943
  Block Number: 1
  Block Time: Fri May 07 2021 13:42:57 GMT+0900 (대한민국 표준시

eth_getTransactionReceipt
eth_getCode
eth_getTransactionByHash
eth_getBlockByNumber
eth_getBalance
eth_getBlockByNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x5235e7cdb366adaf244827a97b20f9031cc2726b37e1
  Gas usage: 42338
  Block Number: 2
  Block Time: Fri May 07 2021 13:42:57 GMT+0900 (대한민국 표준시

eth_getTransactionReceipt
```

8) 스마트 컨트랙트 작성

지금까지는 기본적인 환경에서만 진행을 하였고 이제는 추가로 컨트랙트를 작성하고 다시 compile, migrate 해보려고 한다.

contracts라는 폴더 아래에 새로운 contract 코드를 추가한다.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.8.0;

contract Add {
    uint a=1;
    uint b=2;
```

```
    function addNumber() public {
        a=a+b;
    }

    function getNumber() public view returns(uint) {
        return a;
    }

    function addNumbers(uint x, uint y) public view returns(u
        return x+y;
    }
}
```

필수는 아니나, **<u>contract의 이름과 solidity 파일명은 일단 통일시키자.</u>** (그게 심신에 이롭습니다.)

9) js 파일 추가

migrations 폴더에 보면 1_initial_migration.js라는 파일을 확인할 수 있을 것이다.

이 폴더 아래에 2_add.js라는 파일을 추가시킨다. 그리고 아래와 같이 채워넣는다.

```
// 장소 : migrations 폴더 안 1_add.js
const solidity 파일명 = artifacts.require("solidity 파일명");

module.exports = function (deployer) {
  deployer.deploy(solidity 파일명);
};
```

위와 같이 코드를 추가하고 저장한다

10) compile, migrate 재실행

compile과 migrate은 각각 4-1과 6번 과정이다.

```
// truffle compile
Compiling your contracts...
```

```
============================
> Compiling .\contracts\add.sol
> Artifacts written to C:\Users\Q\Desktop\truffle_tuto\build\
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang

// truffle migrate
Compiling your contracts...
============================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'development'
> Network id:      1620362480211
> Block gas limit: 6721975 (0x6691b7)


2_add.js
========

   Deploying 'add'
   ---------------
   > transaction hash:    0xb98450f062f9bc69c5c0fc3a9c16e1d54
   > Blocks: 0            Seconds: 0
   > contract address:    0xFBdfeC441215265739a09990bE013353a
   > block number:        3
   > block timestamp:     1620363862
   > account:             0xf8a7fD8Df31BECD2a8d25a69302d402a8
   > balance:             99.99338868
   > gas used:            96285 (0x1781d)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0019257 ETH
```

```
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0019257 ETH


Summary
=======
> Total deployments:   1
> Final cost:          0.0019257 ETH

// migrate 후(ganache cli 화면)
eth_blockNumber
net_version
eth_accounts
eth_getBlockByNumber
eth_getCode
eth_getBlockByNumber
eth_call
eth_getBlockByNumber
eth_getCode
eth_getBlockByNumber
eth_call
eth_getBlockByNumber
eth_accounts
net_version
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_getBlockByNumber
eth_estimateGas
net_version
eth_blockNumber
eth_getBlockByNumber
```

```
eth_sendTransaction

  Transaction: 0xb98450f062f9bc69c5c0fc3a9c16e1d54beb464f3ca
  Contract created: 0xfbdfec441215265739a09990be013353ac686b
  Gas usage: 96285
  Block Number: 3
  Block Time: Fri May 07 2021 14:04:22 GMT+0900 (대한민국 표준시

eth_getTransactionReceipt
eth_getCode
eth_getTransactionByHash
eth_getBlockByNumber
eth_getBalance
eth_getBlockByNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x8373434eb6fb72fb142e288ec81998b7d59020f40c0
  Gas usage: 27338
  Block Number: 4
  Block Time: Fri May 07 2021 14:04:22 GMT+0900 (대한민국 표준시

eth_getTransactionReceipt
```

11) 특정 파일만 compile, migrate 하기

특정 파일만 컴파일 하기

```
truffle compile .\contracts\AS.sol

truffle migrate --f 2 (2번부터 끝까지)
truffle migrate --f 2 --to 2 (2번부터 2번까지, 2번만)
```

12) 반복 실행하기

```
var a = 0;
let timer = setInterval(()=>add.setA(a+=1), 2000);
setTimeout(()=>{clearInterval(timer)}, 40000)
```

▼ Truffle console

1. 터미널에 truffle console을 실행시키면 console 창이 실행된다.

```
truffle console
// 아래와 같은 창이 보이게 된다.
truffle(development)>
```

2. deploy가 된 계약을 변수로 설정한다.

```
let add = await Add.deployed()
// 아래와 같은 결과가 보이게 된다.
undefined
```

3. 계약내 특정 함수를 실행한다.

```
add.addNumber()
add.getNumber()
```

4. 특정 함수를 실행하고나면 해당 결과가 거래로 등록되는 것을 확인한다.

▼ Goerli testnet에 contract 배포하기

```
mkdir goerli_testnet
cd goerli_testnet

truffle create contract A
```

```
// contracts/A.sol
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;
```

```
contract AAA {
  uint public a = 100;

  function setA(uint _a) public {
    a = _a;
  }

  function add(uint _a, uint _b) public pure returns(uint) {
    return _a+_b;
  }
}
```

```
goerli_tesnet 아래에 .env 파일 생성
MNEMONIC = 니모닉 코드 넣기
PROJECT_ID = api key 넣기
```

```
//truffle-config.js
//아래 부분 주석 제거
require('dotenv').config();
const { MNEMONIC, PROJECT_ID } = process.env;

const HDWalletProvider = require('@truffle/hdwallet-provider

goerli: {
      provider: () => new HDWalletProvider(MNEMONIC, `https:/
      network_id: 5,       // Goerli's id
      confirmations: 2,    // # of confirmations to wait betv
      timeoutBlocks: 200,  // # of blocks before a deployment
      skipDryRun: true     // Skip dry run before migrations'
    },
```

```
npm install dotenv
```

```
npm i @truffle/hdwallet-provider@next

truffle compile
truffle migrate --network goerli
truffle console --network goerli

let aaa = await AAA.deployed()
aaa.abi
aaa.a()
aaa.setA(50)
aaa.a()
```

▼ constructor 있는 contract 배포하기

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract A {
  uint public a;

  constructor(uint _a) {
    a = _a;
  }

  function setA(uint _a) public {
    a = _a;
  }
}

contract B {
  uint public a;

  function setA(uint _a) public {
    a = _a;
```

```
  }
}
```

```
// 장소 : migrations 폴더 안 1_add.js
const AA = artifacts.require("B");
const AB = artifacts.require("A");

module.exports = function (deployer) {
  deployer.deploy(AA);
  deployer.deploy(AB,12); // 그냥 뒤에 필요한 input값들을 같이 넣어
};
```

trouble shooting

▼ npm 설치

```
sudo npm cache clean -f
sudo npm i -g npm
```

▼ 자세한 내용은 about_Execution_Policies(https://go.microsoft.com/fwlink/?LinkID=135170)를 참조하십시오.

```
PS C:\Users\computer> truffle version
truffle : 이 시스템에서 스크립트를 실행할 수 없으므로 C:\Users\comput
수 없습니다. 자세한 내용은 about_Execution_Policies(https://go.mi
위치 줄:1 문자:1
```

```
Get-ExecutionPolicy
// Restricted로 아마 뜰 것
```

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

```
Get-ExecutionPolicy
// RemoteSigned로 아마 뜰 것
```

```
PS C:\Users\computer> truffle version
Truffle v5.9.4 (core: 5.9.4)
Ganache v7.8.0
Solidity v0.5.16 (solc-js)
Node v18.16.0
Web3.js v1.10.0
```

▼ npx 설치

```
npm i npx -g
```

▼ node 설치

```
1) node -v
2) npm cache clean -f
3) npm install -g n
4) n lts
```

▼ npx

```
npx truffle
npx ganache-cli
```

▼ ganache-cli 설치 관련

```
Compiling your contracts...
===========================
√ Fetching solc version list from solc-bin. Attempt #1
√ Downloading compiler. Attempt #1.
> Everything is up to date, there is nothing to compile.
```

```
> Something went wrong while attempting to connect to the net

Could not connect to your Ethereum client with the following
    - host       > 127.0.0.1
    - port       > 8545
    - network_id > *
Please check that your Ethereum client:
    - is running
    - is accepting RPC connections (i.e., "--rpc" option is u
    - is accessible over the network
    - is properly configured in your Truffle configuration fi

Truffle v5.3.4 (core: 5.3.4)
Node v14.15.1
```

ganache-cli가 실행되고 있지 않은 상태이다.