Getting Started with Ethereum Blockchain Development

Marisa Paryasto
25 Nov 2019

Blockchain Development Environment Installation

- Node Environment. Download node.js installer here https://nodejs.org/en/
- Solc Compiler. npm install -g solc
- Go Ethereum. Download here https://geth.ethereum.org/downloads/
- Truffle Framework. npm install -g truffle
- Ganache. npm install -g ganache-cli
- truffle test

- npm install

- Unlock ganache-cli account

```
truffle console
web3.personal.unlockAccount(firstAccountAdress,firstPriv
ateKey,15000)
```

- Run interaction layer

```
node interaction/interaction.js
```

```
npm install original-require

npm install -g truffle-expect truffle-config web3
```

```
Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
```

```
npm install --force -g truffle
$ truffle init
$ truffle migrate
```

Create truffle project

- initialize project: ***truffle init***
- Following file and folders will be created:
  - **contracts:** Folder to keep solidity contracts
  - **migrations:** JS scripts to deploy solidity contracts
  - **test:** Truffle test cases
  - **truffle.js:** Truffle config file

Writing Smart Contract

1. **Add truffle config file:** Add below config to truffle.js. In below config, we are declaring development network, which will connect ethereum node running on localhost and port 8545.

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*" // Match any network id
    }
  }
};
```

Writing Smart Contract

2. **Creating a contract**: Lets create a simple contract 'counter.sol' file in contract folder. Counter contract defines following things:

I. **Events**:
What are events in Solidity: Events are dispatched signals that smart contracts can fire. DApps, or anything connected to Ethereum JSON-RPC API(like node js application), can listen to these events and act accordingly. Counter contract defines two events-

- CounterIncrementedEvent
- CounterDecrementedEvent

II. **Variable**: Like other programming languages, variable is something that holds data.
III. **Methods**: Counter contract defines three methods to increment, decrement and get counter value.

```
pragma solidity ^0.4.23;
contract Counter {

    event CounterIncrementedEvent(int count);
    event CounterDecrementedEvent(int count);
    int private count = 0;
    function incrementCounter() public {
        count += 1;
        emit CounterIncrementedEvent(count);
    }
    function decrementCounter() public {
        count -= 1;
        emit CounterDecrementedEvent(count);
    }
    function getCount() public constant returns (int) {
        return count;
    }
}
```

Writing Smart Contract

3. **Deployment Script** — Add deployment script named as ‘2_initial_migration.js’ for Counter.sol

```
var Counter = artifacts.require("./
Counter.sol");          1
module.exports = function(deployer) {
  deployer.deploy(Counter);
};
```

Writing Smart Contract

4. **Write Truffle Test cases:** Create test file named as counter_test.js for Counter contract in test folder.

```
const Counter = artifacts.require("Counter");
contract('Counter test', async (accounts) => {
  let instance;
  before(async () => {
    instance = await Counter.deployed();//deploy contract
  });
  it("Initial value of counter should be zero", async ()
=> {
    let count = await instance.getCount.call({from:
accounts[0]});
    assert.equal(count, 0);
  });
});
```

Writing Smart Contract

5. Running test cases

Run ganache-cli: Run below command to run ganache

```
ganache-cli
```

Run truffle tests: In separate Window, run truffle test cases using below command

```
truffle test
```

Writing Smart Contract

5. Add more tests

i. Add test for Increment Counter :

```
it("Should increment counter", async () => {
  await instance.incrementCounter({from: accounts[0]});
  let count = await instance.getCount.call({from: accounts[0]});
  assert.equal(count, 1);
});
```

ii. Add test to verify if event is emitted on Increment Counter:

```
it("Should emit event on increment counter", async () => {
  let reciept = await instance.incrementCounter({from: accounts[0]});
  assert.equal(reciept.logs.length, 1);
  assert.equal(reciept.logs[0].args.count, 2);
});
```