ropsten testnetにERC20トークンをデプロイする

【前提】

npmとpythonが必要

npmはnode.jsのインストールと合わせて、入手が可能

---ubuntu の場合---

$ curl -sL https://deb.nodesource.com/setup\_6.x | sudo -E bash -

$ sudo apt-get install -y nodejs

---macの場合---

【デプロイ方法】

**1. truffleのインストール**

$ npm install -g truffle

**2. トークンのデプロイ用フォルダを作成し、truffleを初期化**

$ mkdir matsumoToken | cd matsumoToken

$ truffle init

これにより、コントラクトのコンパイル及び、ネットワークにマイグレーションするのに必要なファイルやフォルダの最小構成が作成される。



**3. zeppelin solidityとtruffle hdwallet providerのインストール**

$ npm install zeppelin-solidity@1.7.0

$ npm install truffle-hdwallet-provider

truffle hdwallet providerについては、以下参照のこと

<https://github.com/trufflesuite/truffle-hdwallet-provider>

注意1）truffle hdwallet ledgerは、pythonのversionが3.5には対応していないため、python2.X系を使用するようにnpmのconfigをsetしてから、npm installする

$ npm config set python /path/to/python2.7

注意2) truffle hdwallet providerをインストールする過程で、node-gypをインストールするが、ubuntuの場合エラーが発生することがある。

> scrypt@6.0.3 install /home/ubuntu/trunkToken/node\_modules/scrypt

> node-gyp rebuild

gyp ERR! build error

gyp ERR! stack Error: not found: make

gyp ERR! stack at getNotFoundError (/usr/local/lib/node\_modules/npm/node\_modules/which/which.js:13:12)

gyp ERR! stack at F (/usr/local/lib/node\_modules/npm/node\_modules/which/which.js:68:19)

gyp ERR! stack at E (/usr/local/lib/node\_modules/npm/node\_modules/which/which.js:80:29)

gyp ERR! stack at /usr/local/lib/node\_modules/npm/node\_modules/which/which.js:89:16

gyp ERR! stack at /usr/local/lib/node\_modules/npm/node\_modules/which/node\_modules/isexe/index.js:42:5

gyp ERR! stack at /usr/local/lib/node\_modules/npm/node\_modules/which/node\_modules/isexe/mode.js:8:5

gyp ERR! stack at FSReqWrap.oncomplete (fs.js:165:21)

gyp ERR! System Linux 4.4.0-1052-aws

gyp ERR! command "/usr/local/bin/node" "/usr/local/lib/node\_modules/npm/node\_modules/node-gyp/bin/node-gyp.js" "rebuild"

以下２コマンドを実行してから、再度インストールのコードを実施すると上手くいく。

$ sudo apt-get update

$ sudo apt-get install build-essential

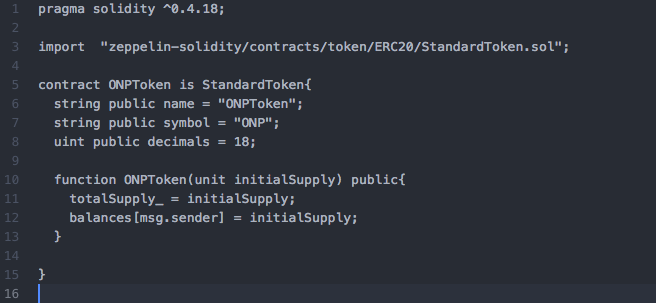
**4. metamaskのニーモニックを取得**

metamaskの登録時に発行されるニーモニック（metamaskを復元する際に使用する秘密鍵）を控えておく。

**5. infuraのアカウントを取得する**

**6. 発行するトークン情報を記述したsolidityファイルの作成**

$vi ./contracts/ONPToken.sol



**7. 発行するトークン情報を記述したjsファイルの作成**

$vi ./migration/2\_deploy\_ONP\_token.js

var TRKToken = artifacts.require("./TRKtoken.sol")

module.exports = (deployer) => {

let initial\_supply = 50000e18

deployer.deploy(TRKToken,initial\_supply)

}

**8. truffle.jsの編集**

$vi truffle.js

# 必要な変数の定義

var HDWalletProvider = require("truffle-hdwallet-provider");

var mnemonic = process.env.ROPSTEN\_MNEMONIC;

var accessToken = process.env.INFURA\_ACCESS\_TOKEN;

# ネットワークの設定

module.exports = {

// See <http://truffleframework.com/docs/advanced/configuration>

// to customize your Truffle configuration!

networks: {

ropsten: {

provider: function() {

return new HDWalletProvider(

mnemonic,

"https://ropsten.infura.io/" + accessToken

);

},

network\_id: 3,

gas: 4500000

}

}

補足1）truffle-hdwallet-providerの３番目の引数にindexを指定すればMetaMaskの他のアドレスを使うことが可能（設定しない場合は一つ目のアカウントを指定）

ex) new HDWalletProvider(mnemonic, "https://ropsten.infura.io/" + accessToken, 3);

補足2)

mnemonic、accessTokenは環境変数から参照できるようにする。

bashを使う場合には、~/.bash\_profileに以下変数を追加する。

export ROPSTEN\_MNEMONIC="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

export INFURA\_ACCESS\_TOKEN="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

**9. コントラクトをコンパイル**

$ truffle compile

-------------------------------------------------------------

matsumototakuyanoMacBook-ea:ONPToken matsumototakuya$ truffle compile --all

Compiling ./contracts/Migrations.sol...

Compiling ./contracts/ONPToken.sol...

Compiling zeppelin-solidity/contracts/math/SafeMath.sol...

Compiling zeppelin-solidity/contracts/token/ERC20/BasicToken.sol...

Compiling zeppelin-solidity/contracts/token/ERC20/ERC20.sol...

Compiling zeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol...

Compiling zeppelin-solidity/contracts/token/ERC20/StandardToken.sol...

Compilation warnings encountered:

/Users/matsumototakuya/Documents/ONPToken/contracts/Migrations.sol:11:3: Warning: Defining constructors as functions with the same name as the contract is deprecated. Use "constructor(...) { ... }" instead.

function Migrations() public {

^ (Relevant source part starts here and spans across multiple lines).

,/Users/matsumototakuya/Documents/ONPToken/contracts/ONPToken.sol:10:3: Warning: Defining constructors as functions with the same name as the contract is deprecated. Use "constructor(...) { ... }" instead.

function ONPToken(uint initialSupply) public{

^ (Relevant source part starts here and spans across multiple lines).

,zeppelin-solidity/contracts/token/ERC20/BasicToken.sol:38:5: Warning: Invoking events without "emit" prefix is deprecated.

Transfer(msg.sender, \_to, \_value);

^-------------------------------^

,zeppelin-solidity/contracts/token/ERC20/StandardToken.sol:33:5: Warning: Invoking events without "emit" prefix is deprecated.

Transfer(\_from, \_to, \_value);

^--------------------------^

,zeppelin-solidity/contracts/token/ERC20/StandardToken.sol:49:5: Warning: Invoking events without "emit" prefix is deprecated.

Approval(msg.sender, \_spender, \_value);

^------------------------------------^

,zeppelin-solidity/contracts/token/ERC20/StandardToken.sol:75:5: Warning: Invoking events without "emit" prefix is deprecated.

Approval(msg.sender, \_spender, allowed[msg.sender][\_spender]);

^-----------------------------------------------------------^

,zeppelin-solidity/contracts/token/ERC20/StandardToken.sol:96:5: Warning: Invoking events without "emit" prefix is deprecated.

Approval(msg.sender, \_spender, allowed[msg.sender][\_spender]);

^-----------------------------------------------------------^

Writing artifacts to ./build/contracts

**10. コンパイルが成功したら、以下コマンドでコントラクトをマイグレート**

$ truffle migrate --network [target\_network]

※　ropstenにマイグレートする場合には、truffle migrate --network ropsten

migrationが成功した場合には、ターミナル上に以下のような表示がされる

--------------------------------------------------

matsumototakuyanoMacBook-ea:ONPToken matsumototakuya$ truffle migrate --network ropsten

Using network 'ropsten'.

Running migration: 1\_initial\_migration.js

Deploying Migrations...

... 0xa9e1782105647527312d2120f0c6d83ecd7cc3fb19e38f91b256ac47d77b9a20

Migrations: 0x95b87d9cbb87442c5952ab2f6ba20b2bc74b446c

Saving successful migration to network...

... 0x36ba066593b874da4d4c8caef1e23ea6122bffd5cab530f7bde412485bfbe33c

Saving artifacts...

Running migration: 2\_deploy\_ONP\_token.js

Deploying ONPToken...

... 0x4e77485af4a9e2ca465a15de928be1a841c8073c9fe6624c58df5f696ec9a4d3

ONPToken: 0xac80c7acb43b356a65ee9cc61ecb6bfac68c07db

Saving successful migration to network...

... 0xb3ce2a4b03c9633ab2ca5d49304f8b776eff908e5e9af4ac75e8bbec0801ced7

Saving artifacts...

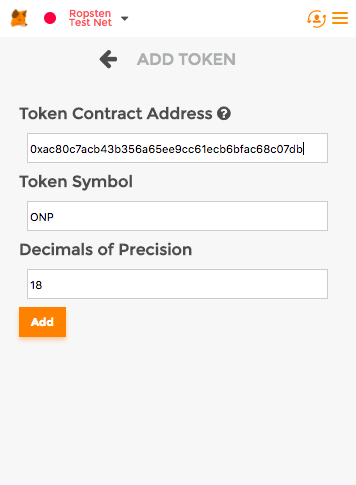
----------------------------------------------------

**11. 発行されたトークンのコントラクトアドレスを確認**

<https://ropsten.etherscan.io/token/0x5f1d239c11ea99f2727c815785bdaff1d12b91f4>

**12. metamaskでトークンを追加する。**

token contract addressに今回発行したトークンのアドレスを入力



**13. 以下のようになれば発行成功**

