

Date : 24/03/2022

Nom : HOSSAIN

Prénom : Ishika

TRAVAUX PRATIQUE :

*Développer, Deloyer et Interagir avec un
contrat intelligent sur Ethereum*

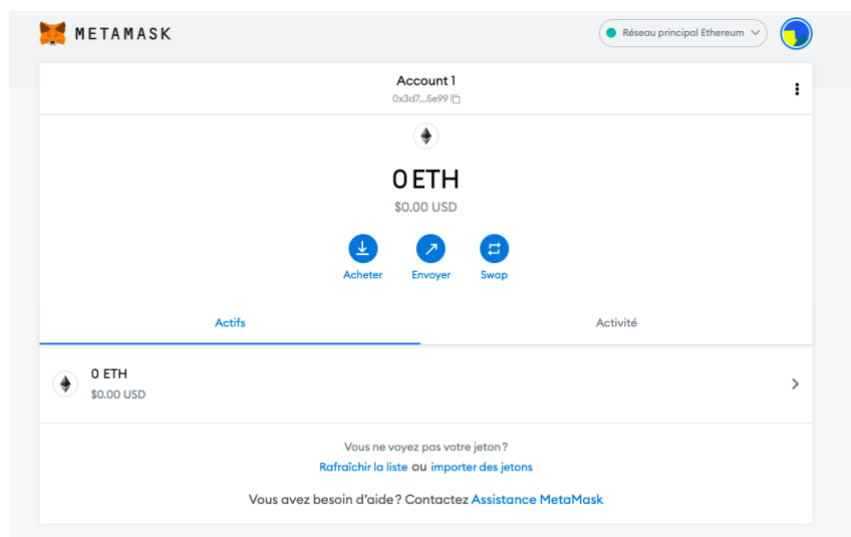
1. Prise en main des outils Remix et Metamask

- a) Téléchargement de Metamask sur le site suivant : <https://metamask.io/>.
- b) Étapes de génération du portefeuille en sauvegardant bien notre seed phase.
- Création d'un nouveau portefeuille



- Création d'un nouveau mot de passe
- Sauvegarde de notre seed

- c) J'ai maintenant accès à mon premier compte « wallet » dont la clé publique commence par « 0x... » (voir illustration ci-dessous).

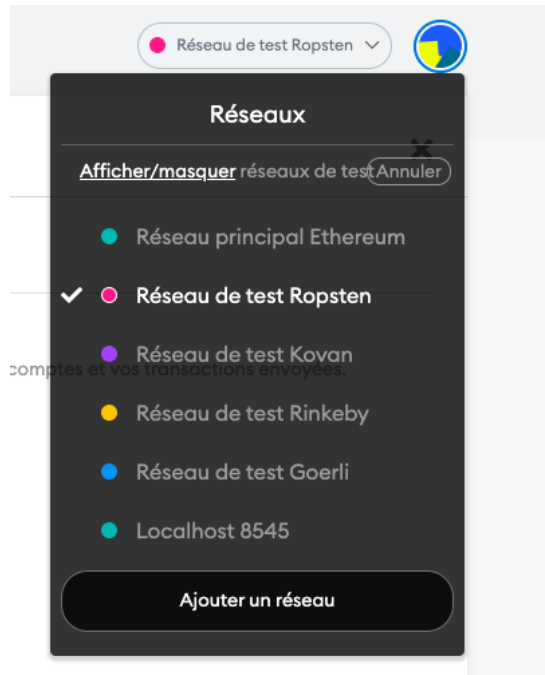


J'active les réseaux de tests, et me connecte sur le réseau de test 'Ropsten'.

Afficher les réseaux de test

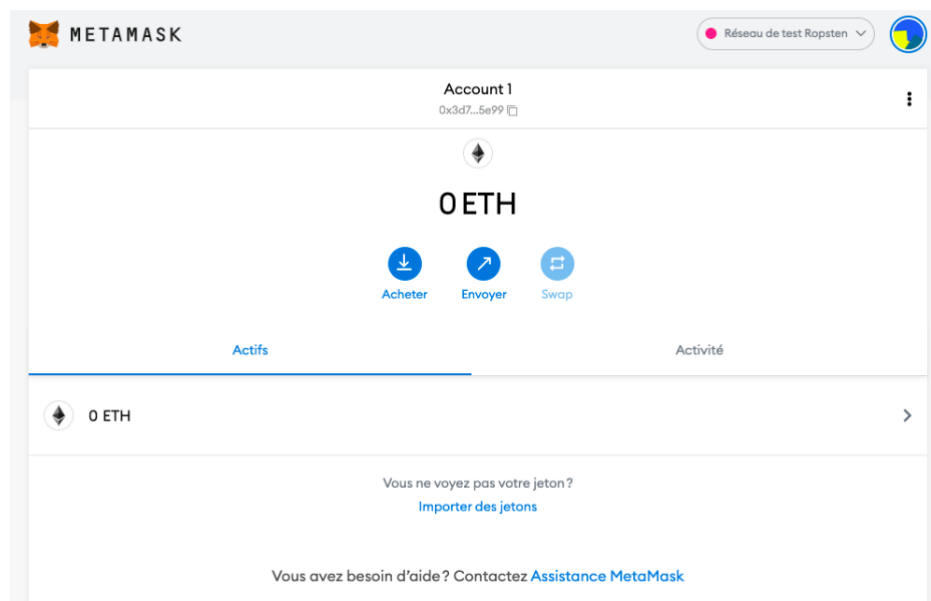
Sélectionnez ceci pour afficher les réseaux de test dans la liste des réseaux

☒ ACTIVÉ



d) Afin de pouvoir réaliser des transactions et déployer un smart contract sur Ethereum il est nécessaire comme vu en cours d'avoir des ETH sur son compte.

- On appuie sur 'Acheter'



- On appuie 'Request 1 ether from faucet'

MetaMask Ether Faucet

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 81144371.40 ether

request 1 ether from faucet

user

address: undefined
balance: ...
donate to faucet:

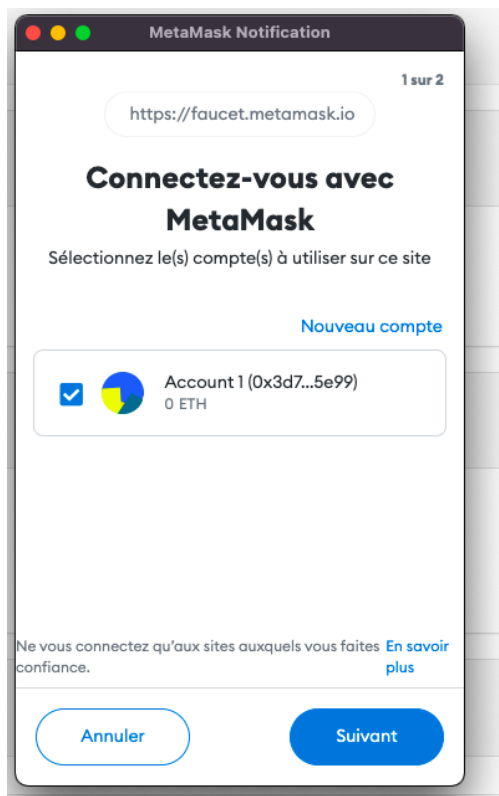
1 ether

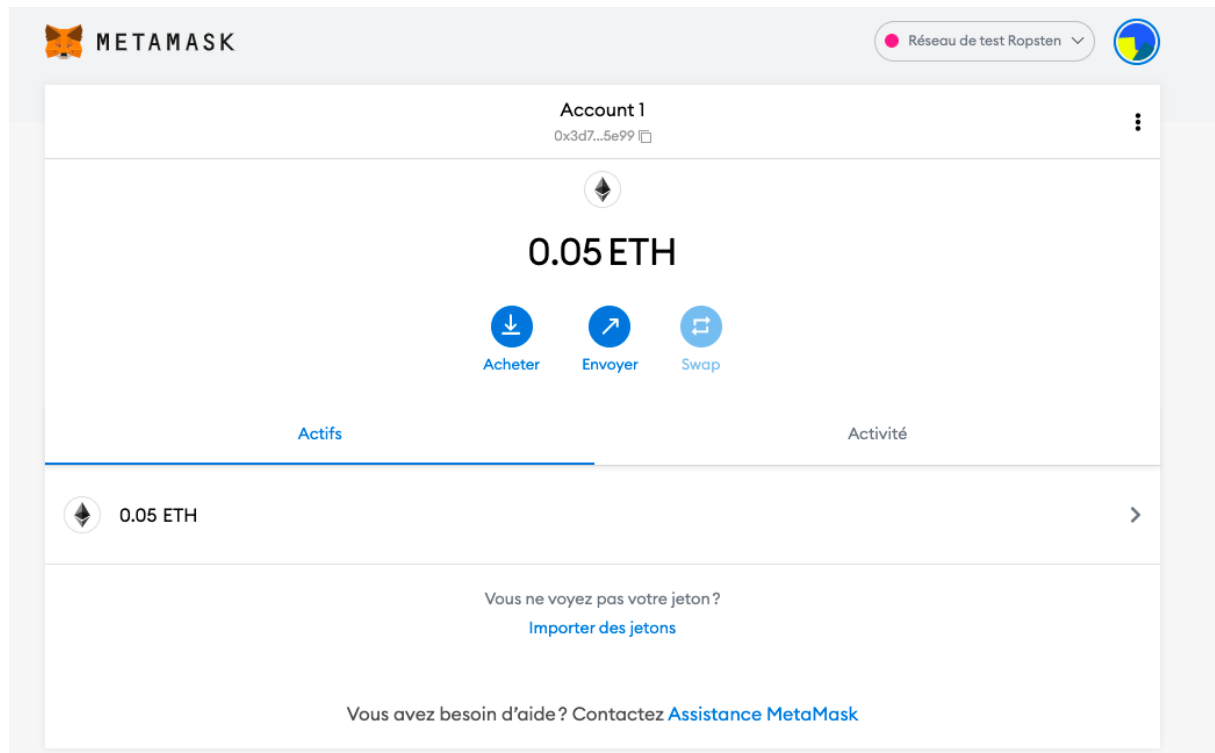
10 ether

100 ether

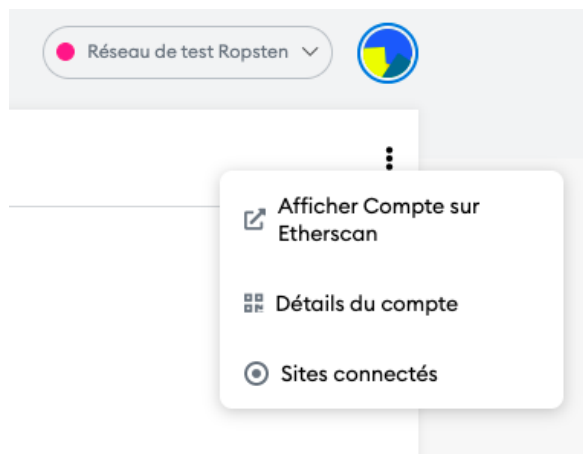
transactions

- On se connecte avec notre compte.





e) Pour consulter la transaction générée vers mon compte, j'appuie sur « Afficher compte sur Etherscan ».



Ensuite, on a accès aux détails de la transaction.

Transaction Details < >

Overview

State

[This is a Ropsten Testnet transaction only]

Transaction Hash: 0x6ad1bfe37ffb3ce8d3e438ed067dc805c7b494097f3cd76e11af2eda132a52c0

Status: Success

Block: 12132628 5 Block Confirmations

Timestamp: 4 mins ago (Mar-25-2022 12:47:36 PM +UTC)

From: 0xb05173465acdd7183e9f32c7a2e1527c840a94f6

To: 0x3d7e2cad60fa81be76421eca3ca9406b7d935e99

Value: 0.05 Ether (\$0.00)

Transaction Fee: 0.001063592145126 Ether (\$0.00)

Gas Price: 0.000000050647245006 Ether (50.647245006 Gwei)

[Click to see More](#) ↓

Gas Price: 0.000000050647245006 Ether (50.647245006 Gwei)

Gas Limit & Usage by Txn: 21,000 | 21,000 (100%)

Gas Fees: Base: 48.222245006 Gwei | Max: 50.783875619 Gwei | Max Priority: 2.425 Gwei

Burnt & Txn Savings Fees: 🔥 Burnt: 0.001012667145126 Ether (\$0.00) 💰 Txn Savings: 0.000002869242873 Ether (\$0.00)

Others: Txn Type: 2 (EIP-1559) Nonce: 0 Position: 7

Input Data: 0x

[Click to see Less](#) ↑




f) On consulte le numéro de Block de notre transaction en appuyant sur le numéro de block.

Block:

12132628

5 Block Confirmations

Voici les détails de la transaction :

Block #12132628	
Overview	
[This is a Ropsten Testnet block only]	
Block Height:	12132628 < >
Timestamp:	8 mins ago (Mar-25-2022 12:48:12 PM +UTC)
Transactions:	0 transaction and 0 contract internal transaction in this block
Mined by:	0x169d07d5c0703733aa505008e9627577c087eb63 in 9 secs
Block Reward:	2 Ether
Uncles Reward:	0
Difficulty:	6,966,893,943
Total Difficulty:	40,887,335,504,537,931
Size:	530 bytes
Gas Used:	0 (0.00%)  -100% Gas Target
Gas Limit:	8,000,000
Base Fee Per Gas:	0.000000018389665301 Ether (18.389665301 Gwei)
Burnt Fees:	 0 Ether
Burnt Fees:	 0 Ether
Extra Data:	Coinfast (Hex:0x436f696e66617374)
Hash:	0xa141421e5fe5373d0f6ca83475a754ef201449999448bf5c74a78d3d5c852b55
Parent Hash:	0x1631e25c14ebff0492df2f5aaa7a9fe0258098e0c720a81cf9e22372f236cc21
Sha3Uncles:	0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
StateRoot:	0x2fee737d3f7ae3257a0f91b03535ce7d8ce760e39d13cc39f075e8ccd12b4eb2
Nonce:	0x461539f057f77e3c
Click to see less ↑	

g) On génère notre première transaction Ethereum sur le réseau Rospden en envoyant 0.01 ETH à l'adresse suivante «0xc25a95A1D4a59A0E56f188f9C966A3Dad518100 » (je n'ai pas 1ETH sur mon compte).

Envoyer

✓

0xc25a95A1D4a59A0E56f188f9C966A3Dad518100

×

F

Nouvelle adresse détectée ! Cliquez ici pour ajouter à votre carnet d'adresses.

Actif:

ETH

Solde: 0.05 ETH

Montant:

0.01 ETH

Aucun taux de conversion disponible

Max.

Annuler

Suivant

< Modifier

Account 1

→

0xc25...100F

Nouvelle adresse détectée ! Cliquez ici pour ajouter à votre carnet d'adresses.

i

New gas experience

We've updated how gas fee estimation and customization works.

Turn on Enhanced Gas Fee UI in Settings

×

ENVOI DE ETH

0.01

MODIFIER

Frais de carburant estimés

0.0007209 0.000721 ETH

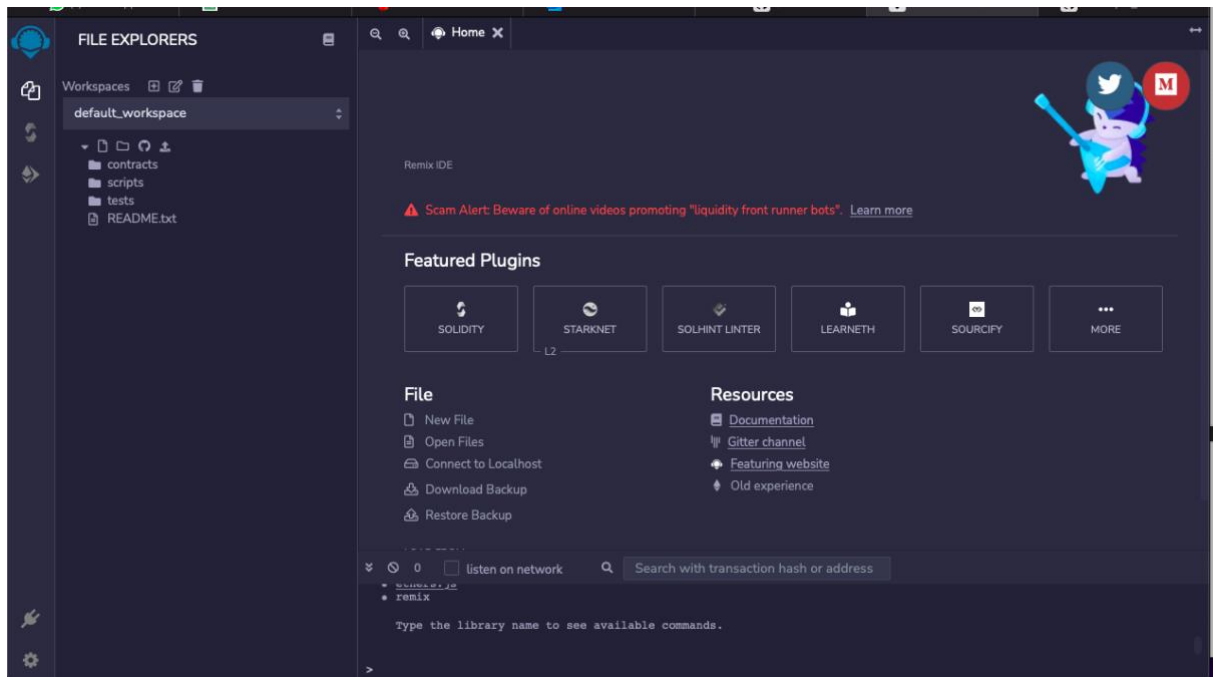
Probablement dans < 30 secondes

Frais maximaux:

0.00080421 ETH

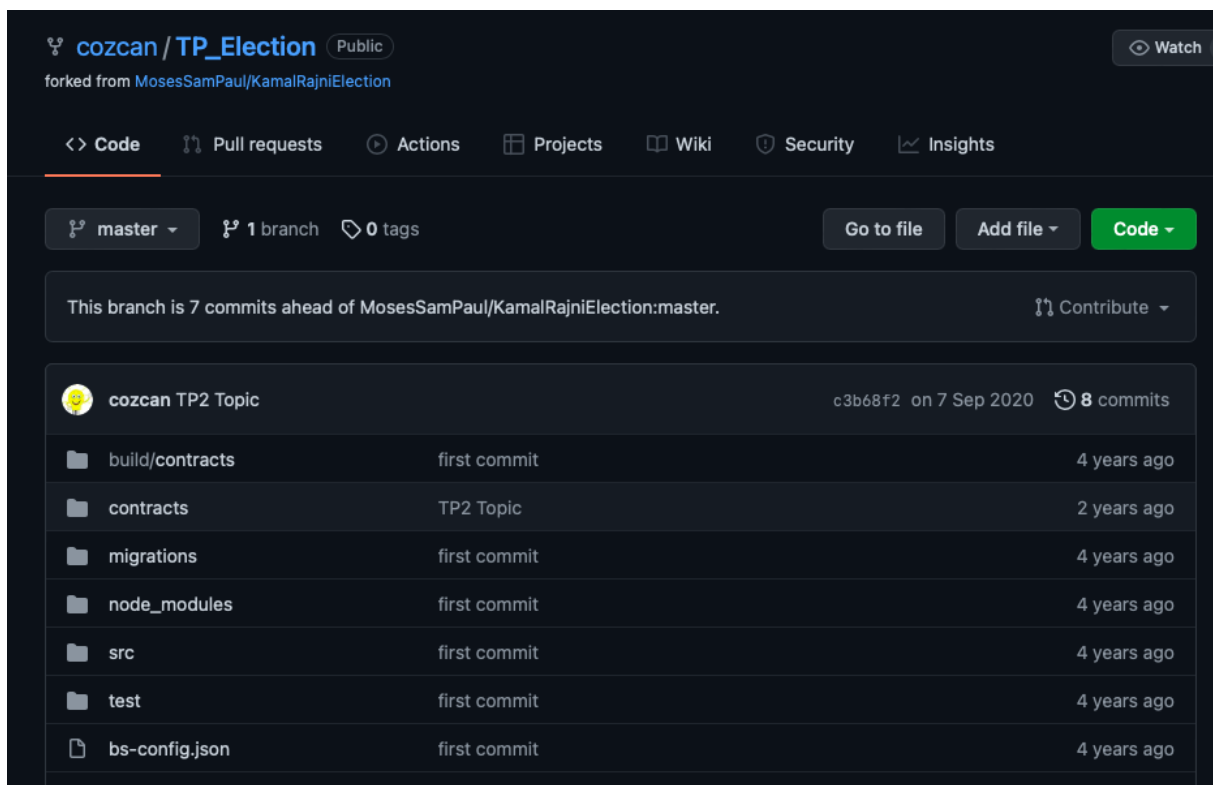
Blockchain 1 - Smart Contract

h) On se rend sur l'IDE Remix.

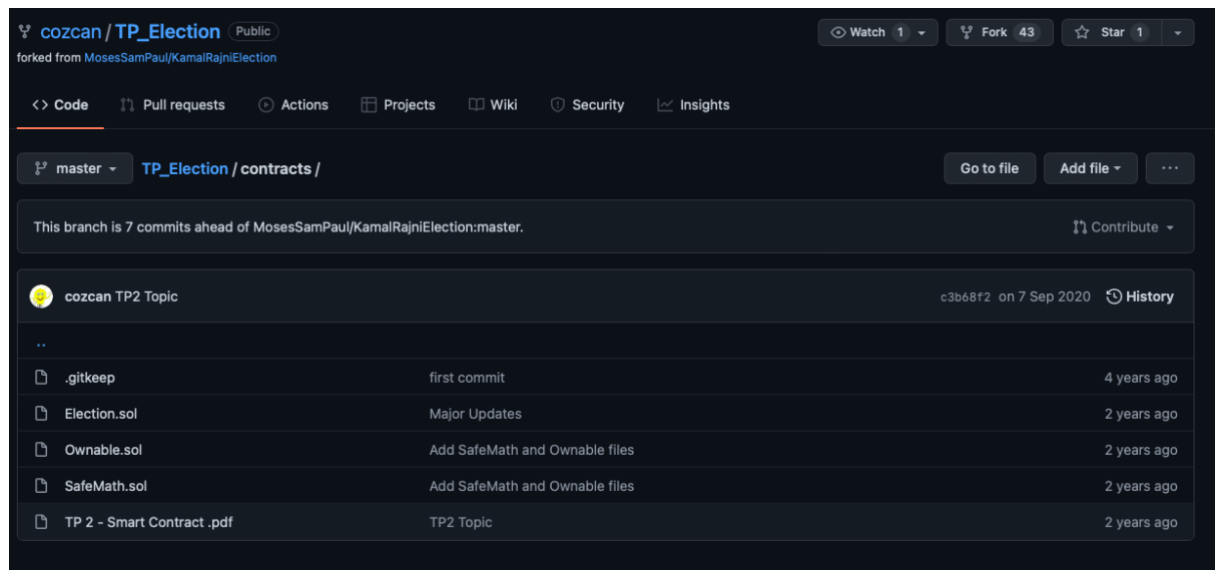


i) On récupère le code source de notre premier smart contract : https://github.com/cozcan/TP_Election

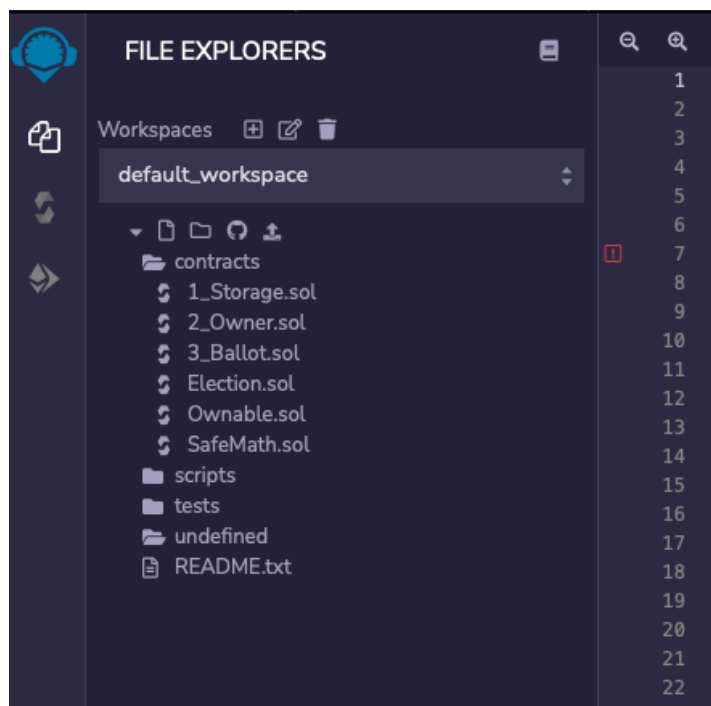
- On se rend dans le folder 'contracts'



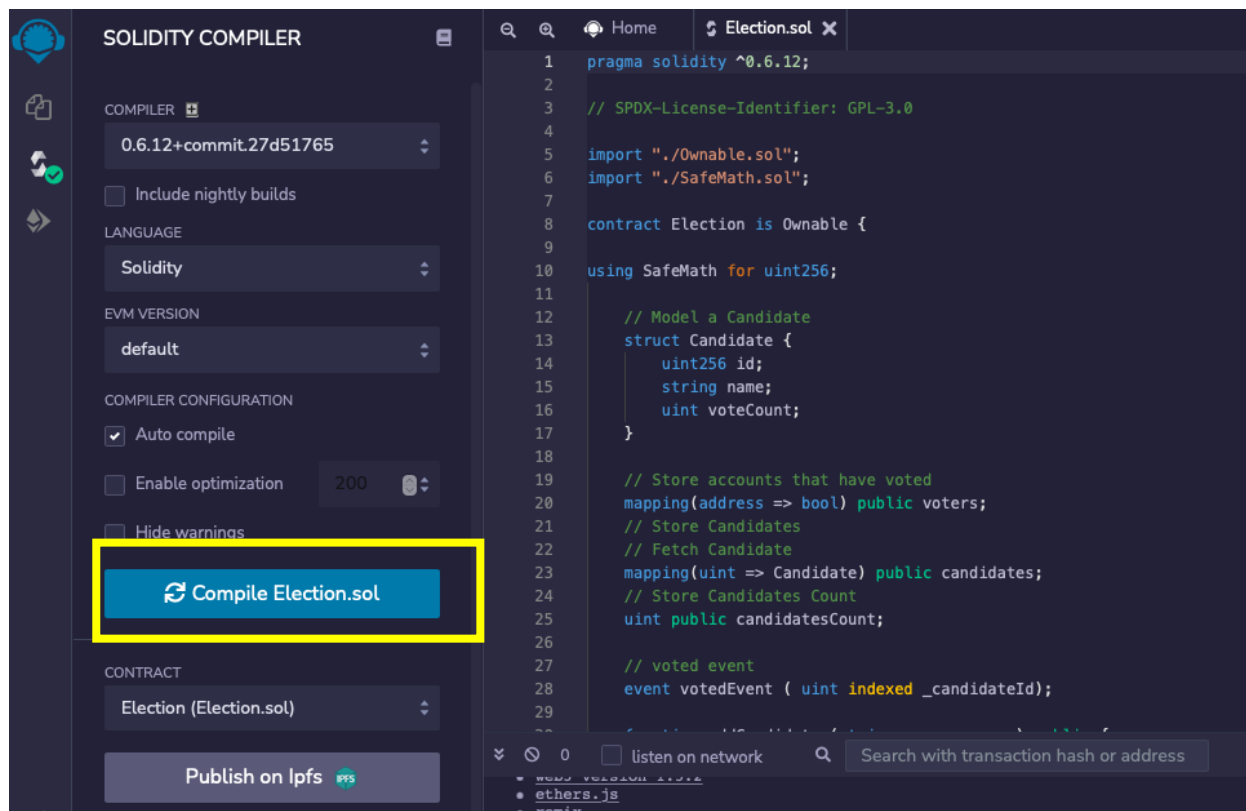
- On télécharge les 3 fichiers .sol



j) On ajoute l'ensemble des fichiers Solidity sur notre environnement Remix.

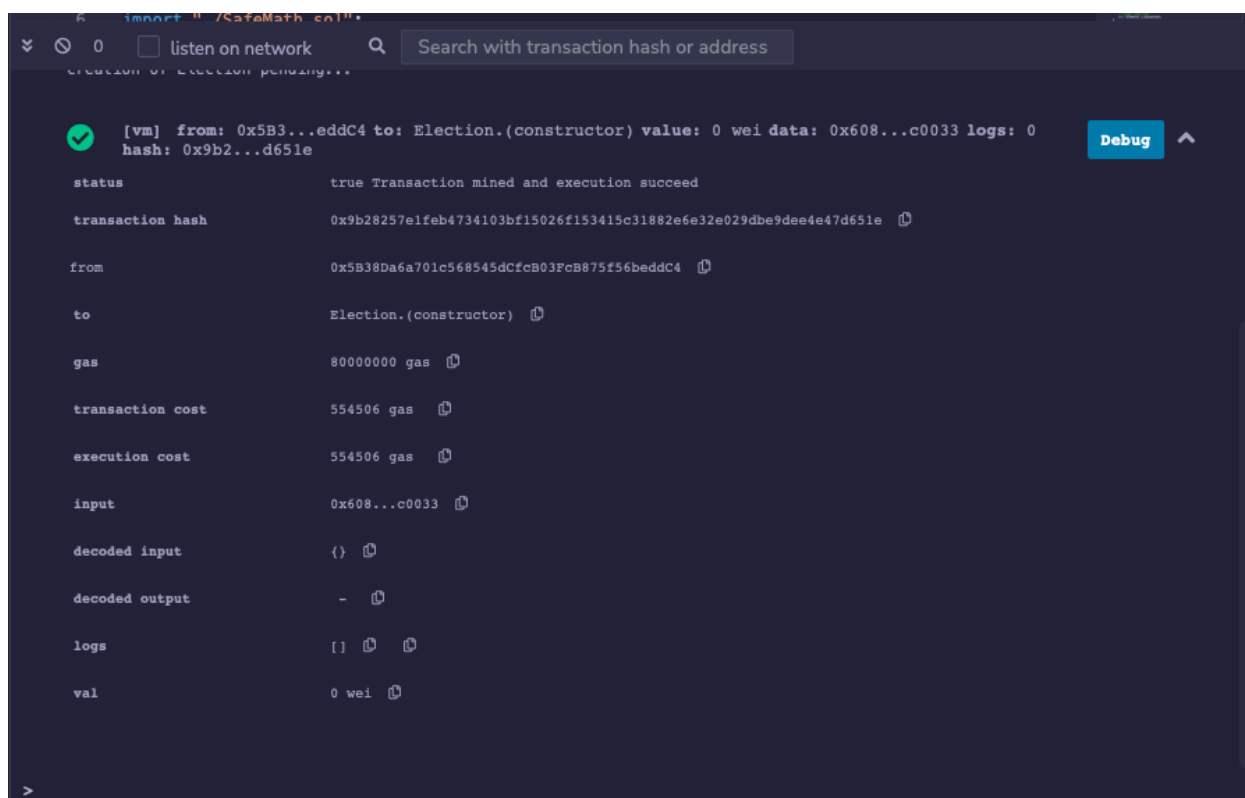


k) On compile notre smart contract « Election » et on fournit l'ABI ainsi que le Byte code du contrat dans des fichiers txt.



L'ABI ainsi que le Byte code du contrat sont présents dans les fichiers txt 'ABI_election.txt' et 'Bytecode.txt'. (Ces fichiers seront sur GitHub)



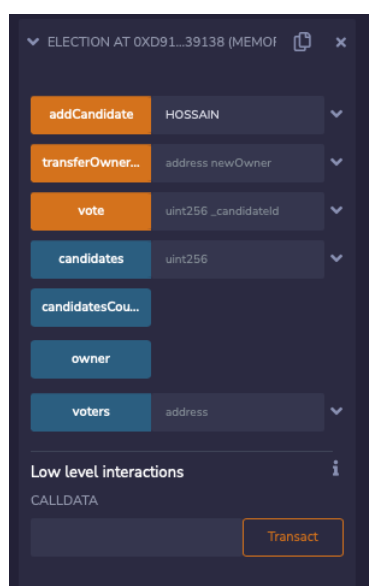


m) Non, les frais de transactions ne sont pas identiques à elle de la transaction sur le TP car le prix du gaz est différent.

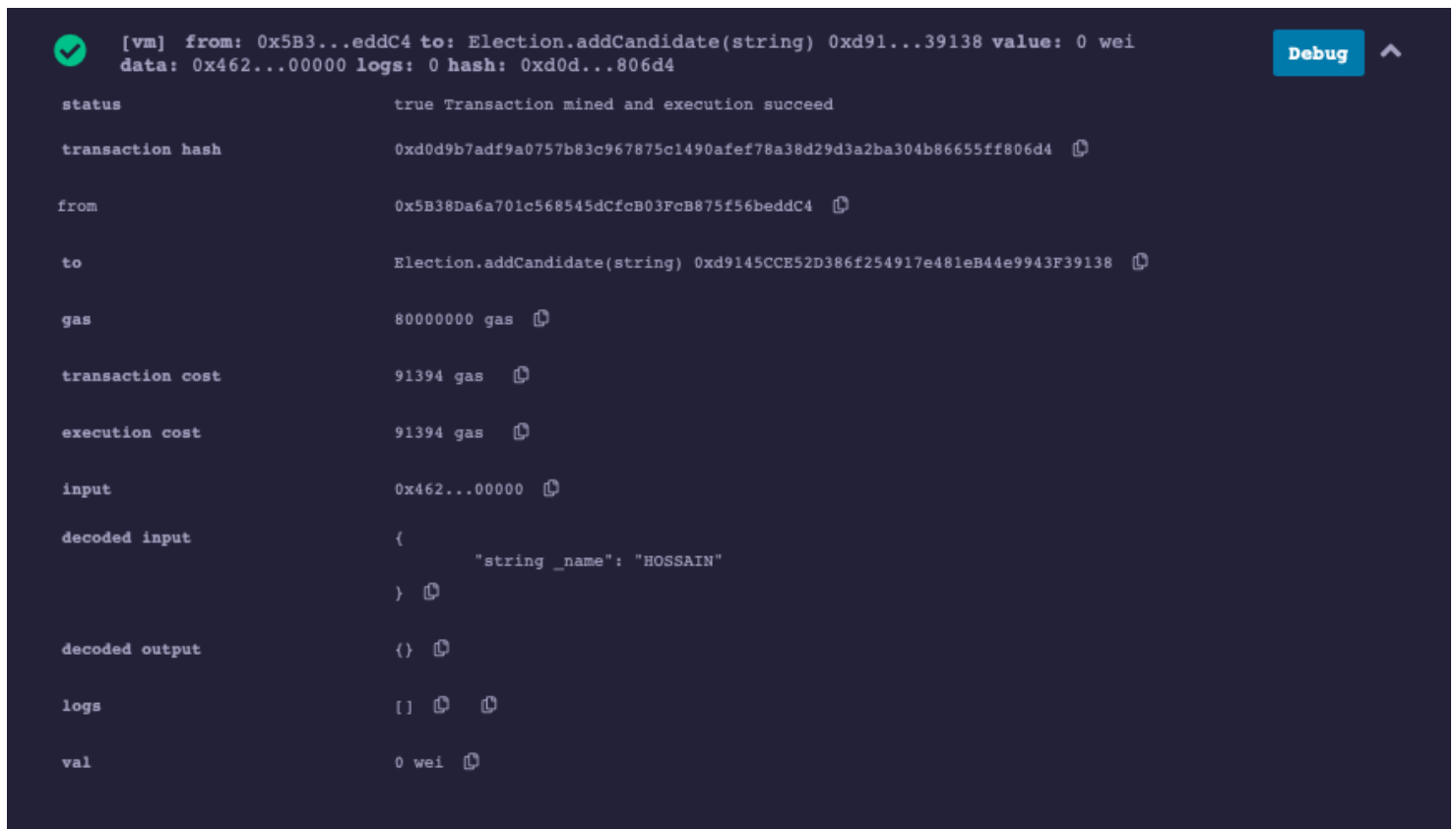
L'adresse public de notre smart contract est :

0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

n) On ajoute le nom du premier candidat en remplaçant par « HOSSAIN ». Et on appuie sur AddCandidate.



- o) Suite à l'ajout du premier candidat, on génère une transaction. On fournit les détails de la transaction :



[vm] from: 0x5B3...eddC4 to: Election.addCandidate(string) 0xd91...39138 value: 0 wei
data: 0x462...00000 logs: 0 hash: 0xd0d...806d4

status true Transaction mined and execution succeed

transaction hash 0xd0d9b7adf9a0757b83c967875c1490afef78a38d29d3a2ba304b86655ff806d4

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Election.addCandidate(string) 0xd9145CCE52D386f254917e481eB44e9943F39138

gas 80000000 gas

transaction cost 91394 gas

execution cost 91394 gas

input 0x462...00000

decoded input {
 "string_name": "HOSSAIN"
}

decoded output {}

logs []

val 0 wei

- p) Consulter la valeur de votre CandidateID à l'aide de Remix et fournissez le détail.

On remarque dans le fichier election.sol, la fonction addCandidate, qui est une fonction d'incrément par 1. C'est-à-dire, qu'à chaque valeur de CandidateID qu'on va rajouter, la valeur va s'incrémenter.

```
function addCandidate (string memory _name) public {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
```

```
{
    "string_name": "HOSSAIN"
}
```

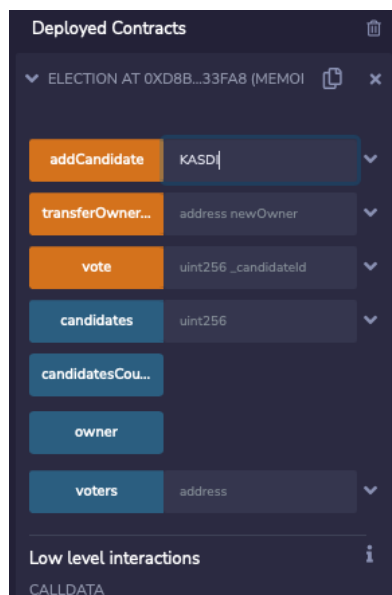
La valeur du premier candidat est '1' :

```

decoded output      {
                    "0": "uint256: id 1",
                    "1": "string: name HOSSAIN",
                    "2": "uint256: voteCount 0"
                    }
logs                []

```

q) Ici, j'ajoute un deuxième candidat avec comme nom « KASDI » dans le smart contract.



Voici les détails de la transaction :

✓
[vm] from: 0x5B3...eddC4 to: Election.addCandidate(string) 0xd8b...33fa8 value: 0 wei data: 0x462...00000 logs: 0 hash: 0x1ee...43f77
Debug
^

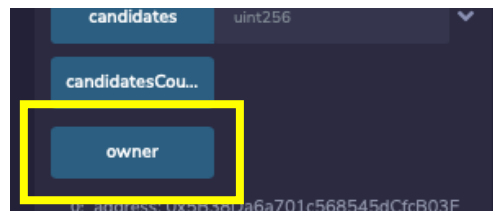
status	true Transaction mined and execution succeed
transaction hash	0x1ee74f2d9f3cdd6d26f808bf38bad5d3c08517cb62a2220bea81682fbfd43f77
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	Election.addCandidate(string) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas	80000000 gas
transaction cost	74270 gas
execution cost	74270 gas
input	0x462...00000
decoded input	<pre>{ "string_name": "KASDI" }</pre>
decoded output	<pre>{}</pre>
logs	<pre>[]</pre>
val	0 wei

r) Voici la valeur du deuxième Candidat. L'Id est bien de 2 :

```
0: uint256: id 2
1: string: name KASDI
2: uint256: voteCount 0
```

s) L'adresse du propriétaire du contract est :
0x5B38Da6a701c568545dCfcB03FcB875f56beddC4.

- On appuie sur Owner



- Voici l'adresse

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Election.owner()
data: 0x8da...5cb5b

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Election.owner() 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8

execution cost 23430 gas (Cost only applies when called by a contract)

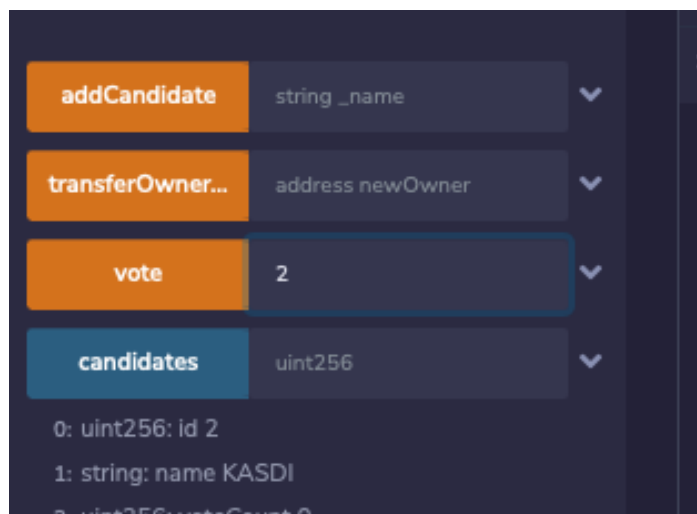
input 0x8da...5cb5b

decoded input {}

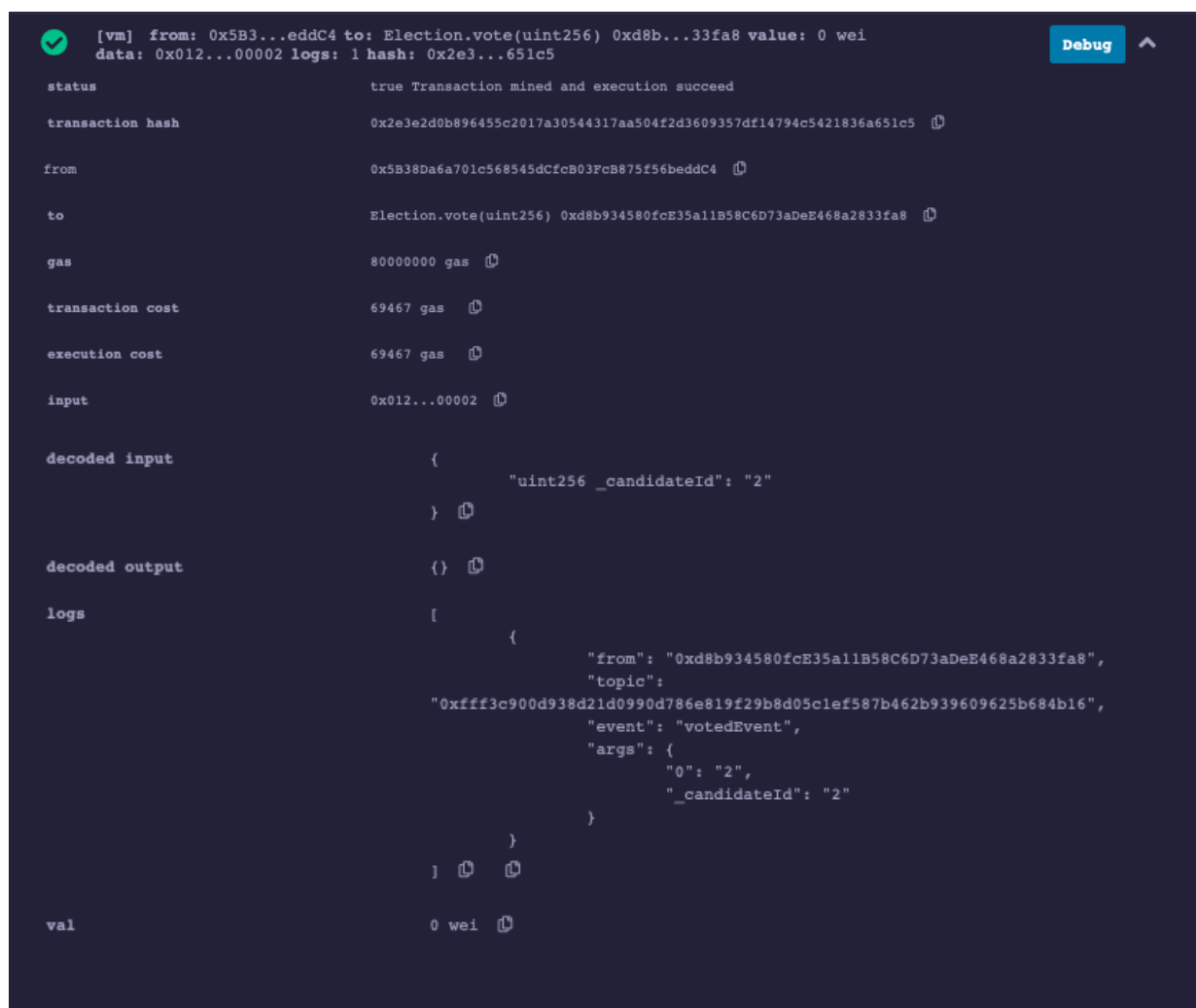
decoded output {
 "0": "address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
}

logs []

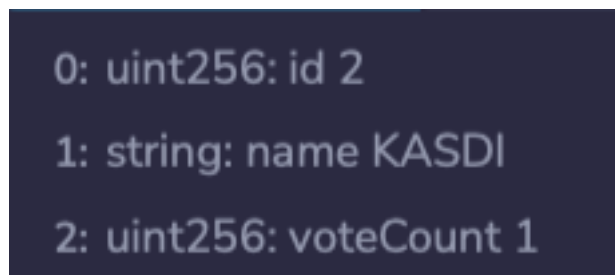
t) On réalise le premier vote pour l'un des candidats à travers Remix.



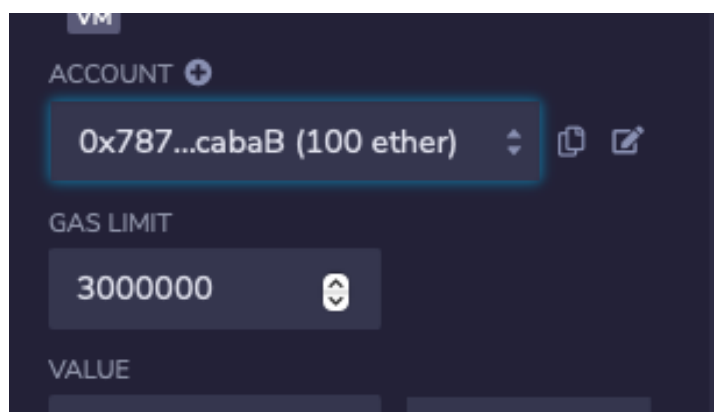
Voici les détails de la transaction :



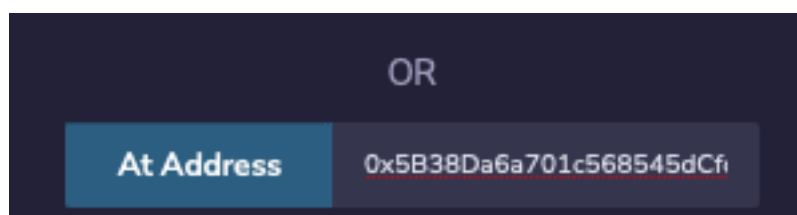
u) Notre vote a bien été prise en compte. Voici la donnée du nombre de vote pour notre candidat :



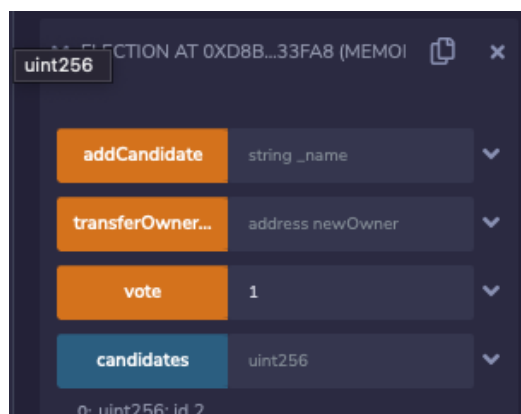
v) Puisque je travaille en local, pour faire cette interaction, je change de compte :



On met l'adresse publique de notre smart contract :



Et on réalise le vote :



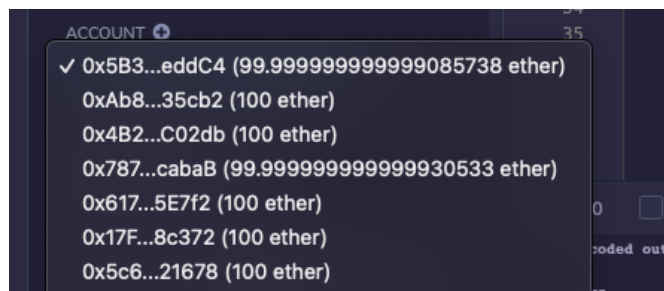
```

status      true Transaction mined and execution succeed
transaction hash  0x47f4fb1b5bc5a24be732ce02c3dff33c048b900da9da4e12df98f4cac99d951f
from        0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB
to         Election.vote(uint256) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas        80000000 gas
transaction cost  69467 gas
execution cost   69467 gas
input         0x012...00001
decoded input   {
                "uint256 _candidateId": "1"
            }
decoded output  {}
logs          [
                {
                    "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                    "topic": "0xfff3c900d938d21d0990d786e819f29b8d05clef587b462b939609625b684b16",
                    "event": "votedEvent",
                    "args": {
                        "0": "1",
                        "_candidateId": "1"
                    }
                }
            ]

```

w) Réaliser ensuite le transfert de la propriété à votre camarade en lui demandant son adresse publique.

- On remet le propriétaire de base



- On met l'adresse publique du second compte dans le « transferOwnership »

- On appuie sur transferOwnership.

```

status      true Transaction mined and execution succeed
transaction hash  0x5ca3ac81f2a418df808da6cadc9e9eld415d7ef5d8ebc708341cfdd14f4e4652
from        0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to          Election.transferOwnership(address) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas         80000000 gas
transaction cost 28682 gas
execution cost  28682 gas
input        0xf2f...cabab
decoded input {
  "address newOwner": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB"
}
decoded output {}
logs         [
logs         [
  {
    "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
    "topic": "0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0",
    "event": "OwnershipTransferred",
    "args": {
      "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
      "1": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
      "previousOwner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
      "newOwner": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB"
    }
  }
]
val         0 wei

transact to Election.transferOwnership pending ...

transact to Election.transferOwnership errored: VM error: revert.

revert
  The transaction has been reverted to the initial state.
Reason provided by the contract: "Not authorized operation".

```

x) Afin de sécuriser l'appel de la fonction addCandidate pour être le seul à pouvoir gérer les candidats, il faudrait vérifier que celui qui ajoute un candidat est le propriétaire du contrat. On va donc limiter cette fonction avec « onlyOwner » et aussi, renvoyer un message d'erreur si ce n'est pas le cas.

y) On modifie le code afin de faire en sorte que vous soyons uniquement le seul à pouvoir ajouter un nouveau candidat :

```

function addCandidate (string memory _name) public onlyOwner {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    require(msg.sender == owner, "Not authorized operation");
}

```