

TRAVAUX PRATIQUE

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

1. Prise en main des outils Remix et Metamask

J'ai créé le repository Système distribué Blockchain et Contrat Intelligent sur Github

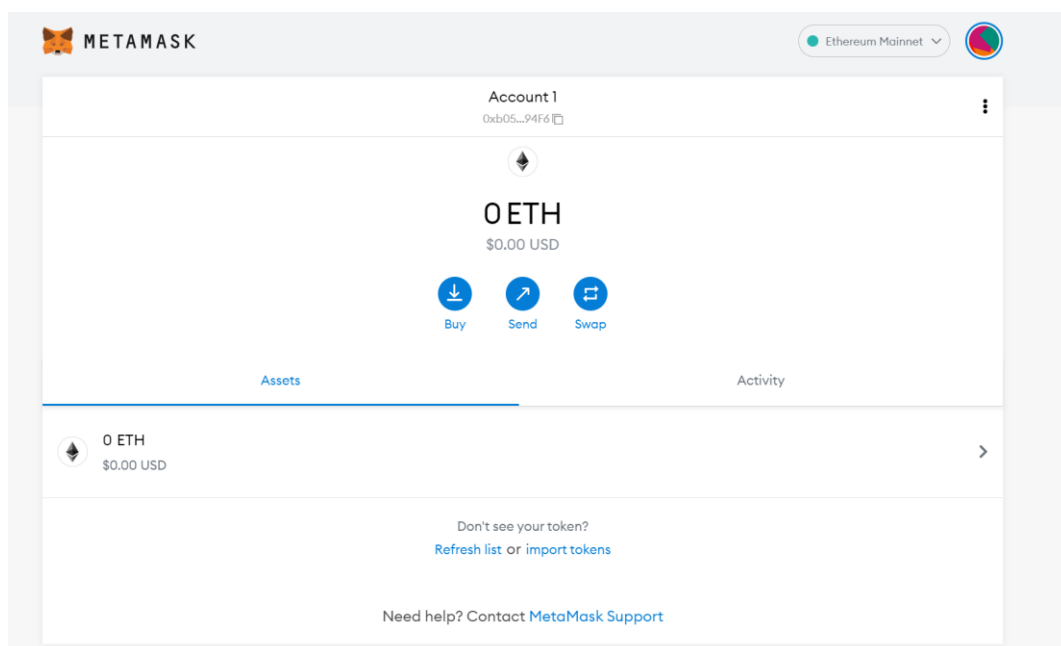
[Systeme-distribue-Blockchain-et-Contrat-intelligent](#) Public

Updated 2 minutes ago

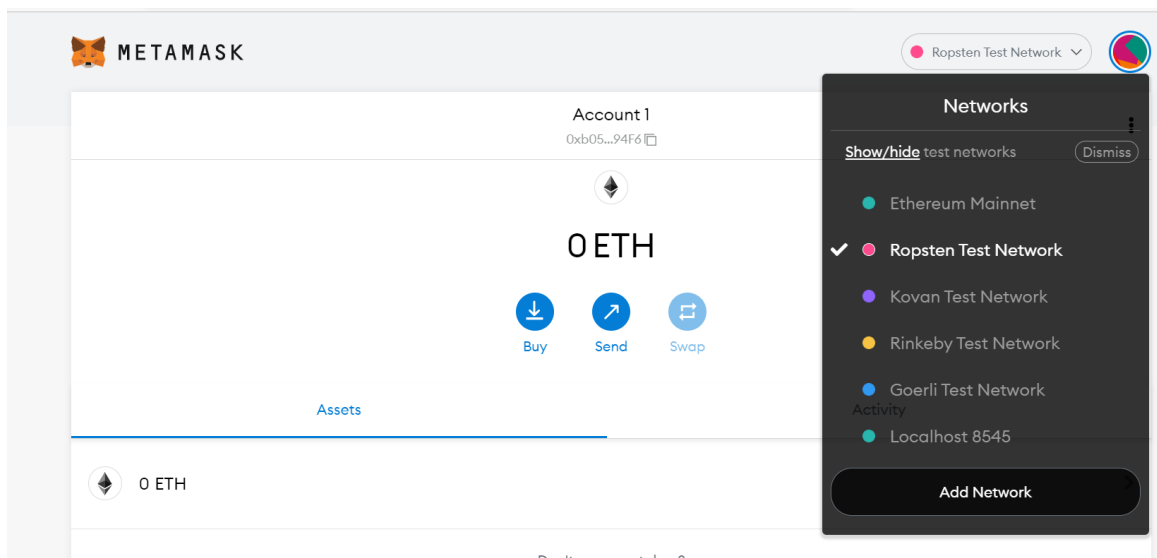
☆ Star ▼

a et b) J'ai téléchargé Metamask en suivant le lien dans l'énoncé du TP.

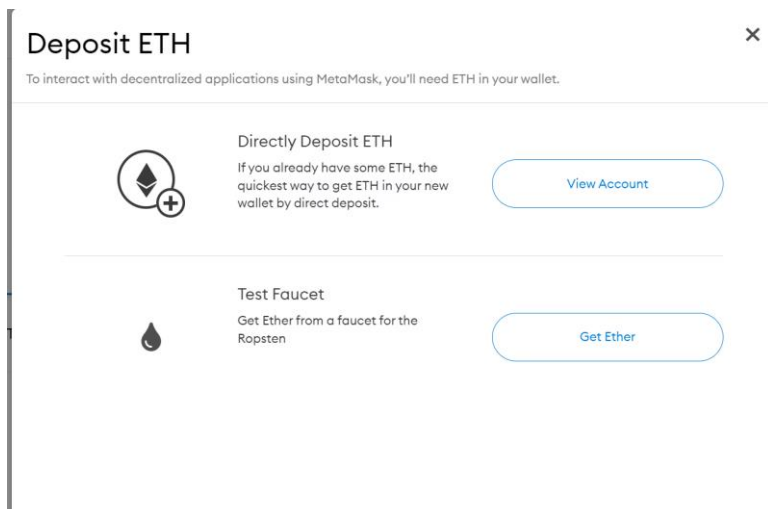
c) J'obtiens mon compte « wallet » dont la clé publique commence par « 0x... ».



Il faut ensuite importer le réseau Ropsten car il n'est pas ajouté dans la configuration de base.



d) Je souhaite recevoir des Ether donc je clique sur « Buy » puis je choisis ensuite Test Faucet et « Get Ether ».



On clique ensuite sur « request 1 ether from faucet » mais une erreur est apparue.

MetaMask Ether Faucet

faucet

address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 81144371.40 ether

[request 1 ether from faucet](#)

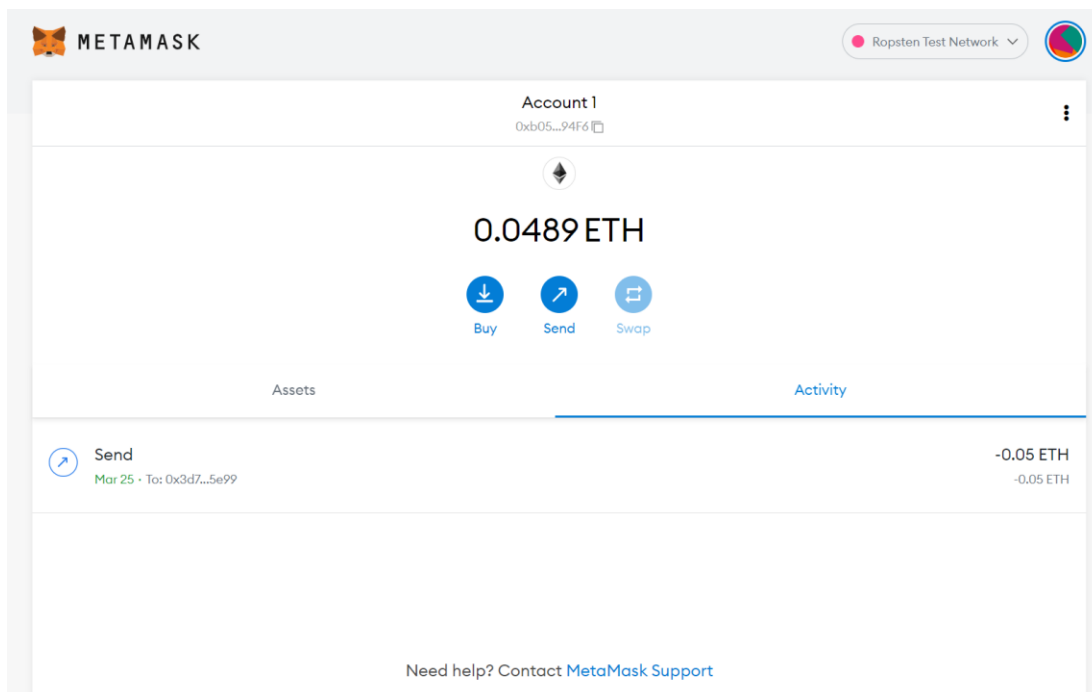
user

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

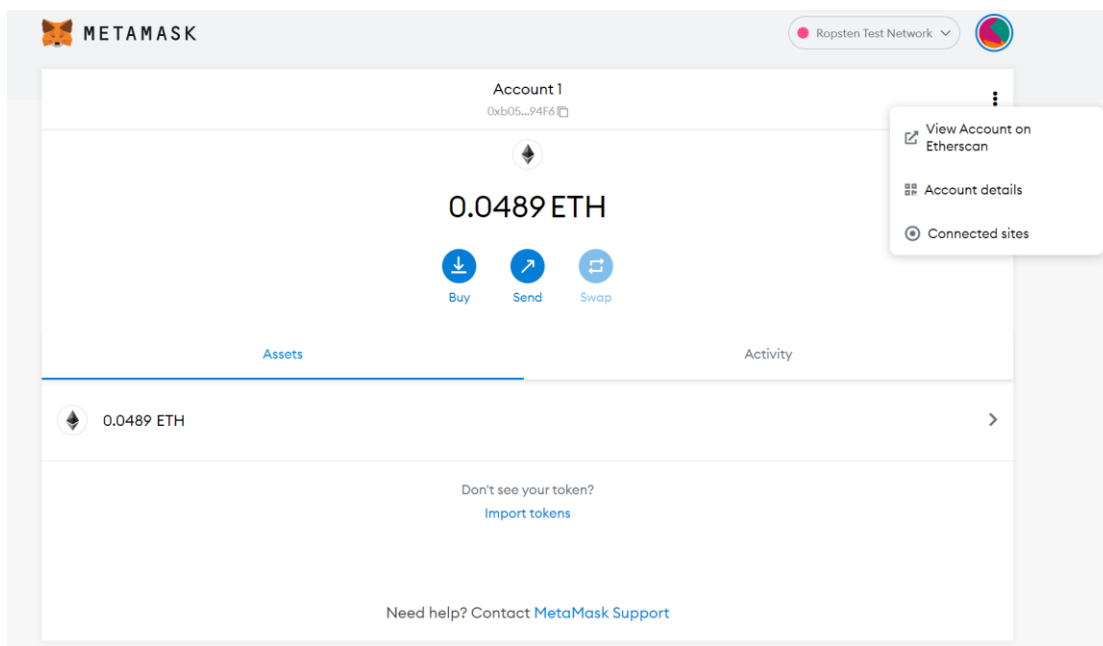
[1 ether](#) [10 ether](#) [100 ether](#)

transactions

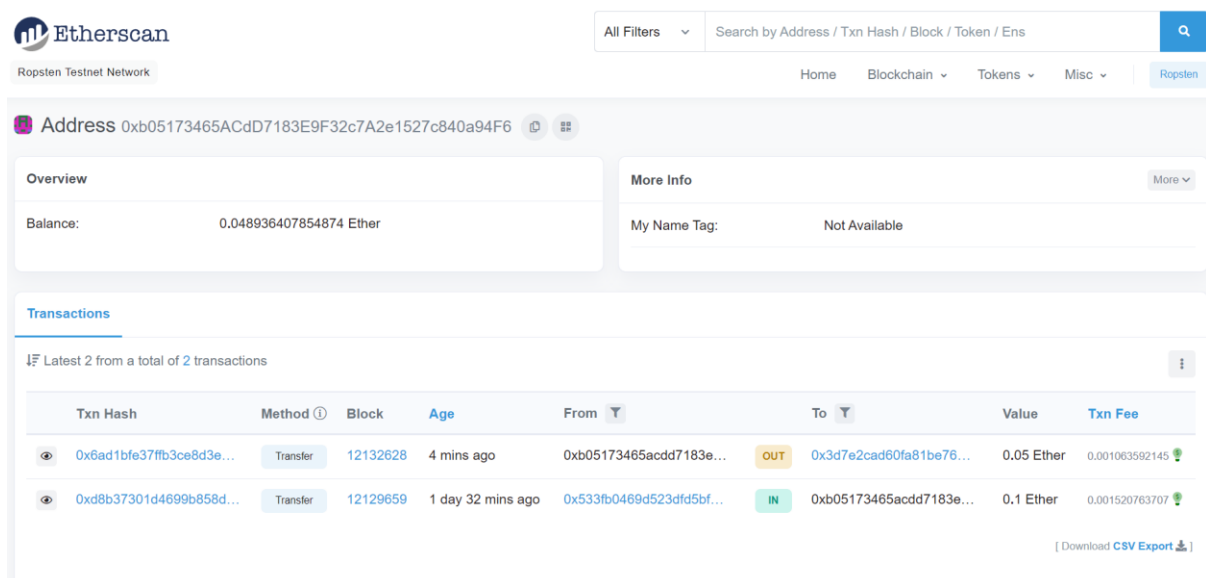
Le prof a donc envoyé 0.1 Ether sur mon numéro de compte (j'ai ensuite envoyé 0.05 à ma camarade qui n'avait rien reçu)



e) Je vais ensuite regarder les détails de la transaction , pour cela je clique sur « View Account on Etherscan »



Je clique ensuite sur la transaction commençant par « 0x53.... »



Voici les détails de la transaction :

Transaction Details < >

Overview State ⓘ

[This is a Ropsten Testnet transaction only]

Transaction Hash:

0xd8b37301d4699b858d6cd8b9f965069589fc7bee31ab78aa50afdb1a2d29c7 ⓘ

Status:

Success

Block:

12129659 2975 Block Confirmations

Timestamp:

⌚ 1 day 33 mins ago (Mar-24-2022 12:20:15 PM +UTC)

From:

0x533fb0469d523dfd5b3d97e0ad75ea66328d08e ⓘ

To:

0xb05173485acd7183e9f32c7a2e1527c840a94f6 ⓘ

Value:

0.1 Ether (\$0.00)

Transaction Fee:

0.001520763707127 Ether (\$0.00)

Gas Price:

0.000000072417319387 Ether (72.417319387 Gwei)

Click to see More ↓

Value:

0.1 Ether (\$0.00)

Transaction Fee:

0.001520763707127 Ether (\$0.00)

Gas Price:

0.000000072417319387 Ether (72.417319387 Gwei)

Gas Limit & Usage by Txn:

21,000 | 21,000 (100%)

Gas Fees:

Base: 0.074391666 Gwei | Max: 72.417319387 Gwei | Max Priority: 72.417254732 Gwei

Burnt & Txn Savings Fees:

🔥 Burnt: 0.000001562224986 Ether (\$0.00)
💰 Txn Savings: 0 Ether (\$0.00)

Others:

Txn Type: 2 (EIP-1559) | Nonce: 80 | Position: 17

Input Data:

0x

Click to see Less ↑

ⓘ A transaction is a cryptographically signed instruction from an account that changes the state of the blockchain. Block explorers track the details of all transactions in the network. Learn more about transactions in our Knowledge Base.

f) Je consulte ensuite le numéro block de ma transaction :

Etherscan

All Filters Search by Address / Txn Hash / Block / Token / ENS

Ropsten Testnet Network

Home Blockchain Tokens Misc Ropsten

Block #12129659

Overview

[This is a Ropsten Testnet block only]

Block Height:

12129659 < >

Timestamp:

⌚ 1 day 35 mins ago (Mar-24-2022 12:20:15 PM +UTC)

Transactions:

19 transactions and 51 contract internal transactions in this block

Mined by:

0x169d07d5c0703733aa505008e9627577c087eb63 in 243 secs

Block Reward:

2.739814702713073759 Ether (2 + 0.740409527390051525 - 0.000594824676977766)

Uncles Reward:

0

Difficulty:



27,642,250,706

Total Difficulty:

40,853,805,187,136,500


Size:

16,052 bytes

⑦ Size:	16,052 bytes
⑦ Gas Used:	7,995,851 (99.95%)  +100% Gas Target
⑦ Gas Limit:	8,000,000
⑦ Base Fee Per Gas:	0.00000000074391666 Ether (0.074391666 Gwei)
⑦ Burnt Fees:	 0.000594824676977766 Ether
⑦ Extra Data:	Coinfast (Hex:0x436f696e66617374)
Click to see more ↓	


g) Je génère ma première transaction Ethereum sur le réseau Rospfen en envoyant 0.01 ETH à l'adresse suivante « 0xc25a95A1D4a59A0E56f188f9C966A3Dad518100F »

Send


0xc25a95A1D4a59A0E56f188f9C966A3Dad518100F

New address detected! Click here to add to your address book.

Asset:


ETH
Balance: 0.04976704 ETH

Amount:

0.01 ETH
No Conversion Rate Available

Max

Cancel

Next

[Edit](#)

Account 1

→

0xc25...100F

New address detected! Click here to add to your address book.

SENDING ETH

0.01

Estimated gas fee

0.00027744 0.000277 ETH

Likely in < 30 seconds

Max fee: 0.00033599 ETH

Total

0.01027744 0.01027744 ETH

Amount + gas fee

Max amount: 0.01033599 ETH

Reject

Confirm

Queue (1)

↗

Send

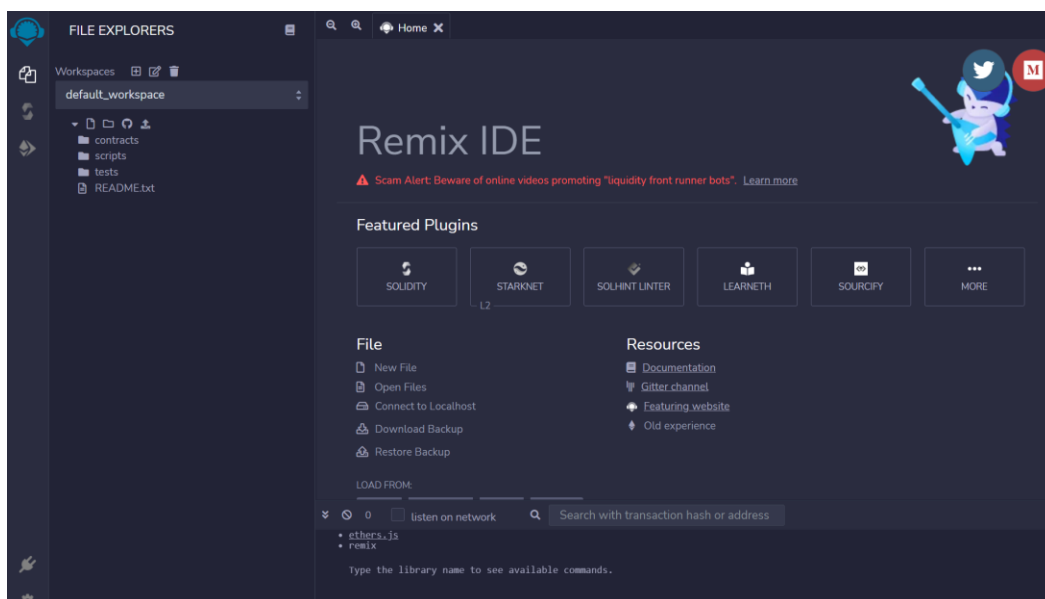
Pending · To: 0xc25...100f

Speed Up

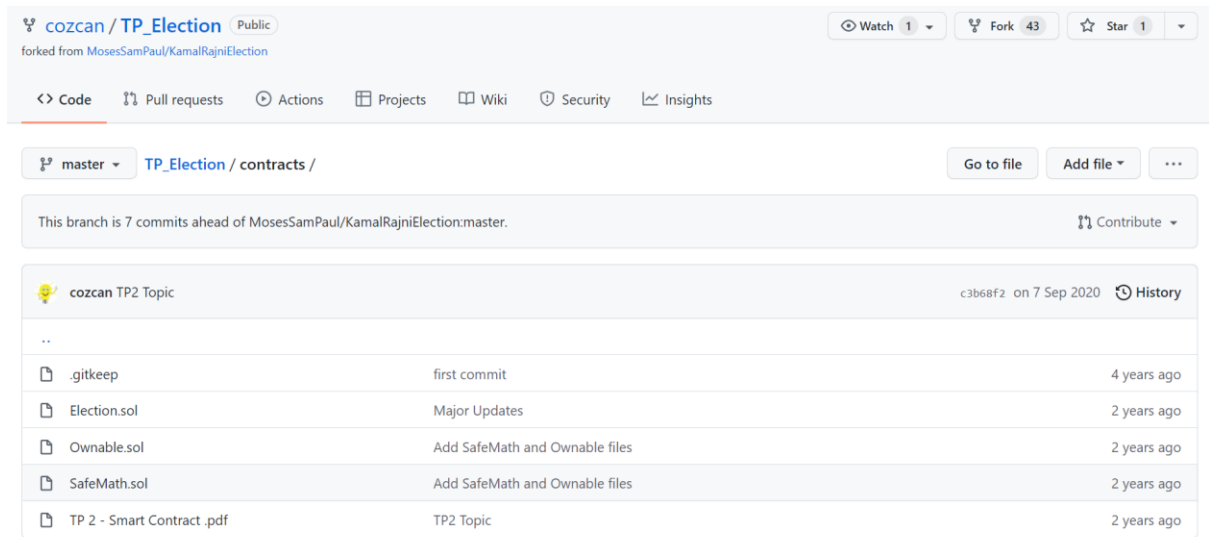
Cancel

-0.01 ETH
-0.01 ETH

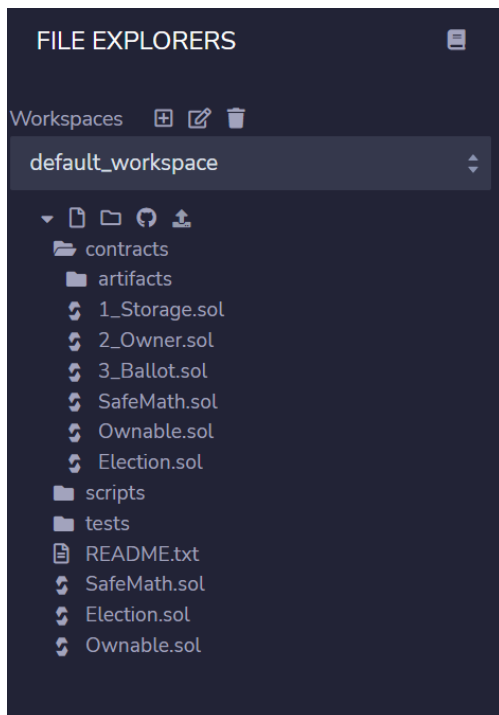
h) Voici l'IDE Remix : remix.ethereum.org



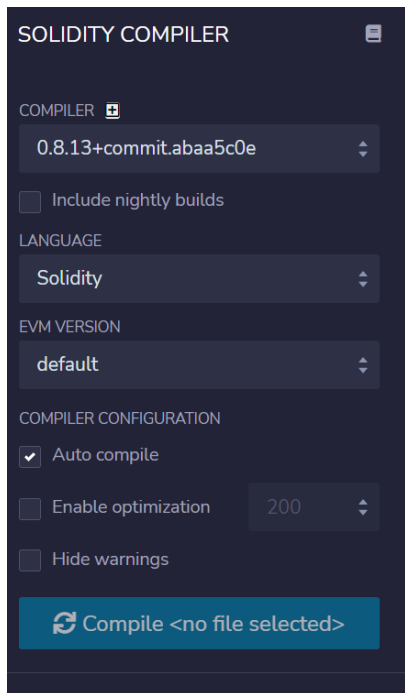
i) Je récupère le code source de mon premier smart contract sur le Github



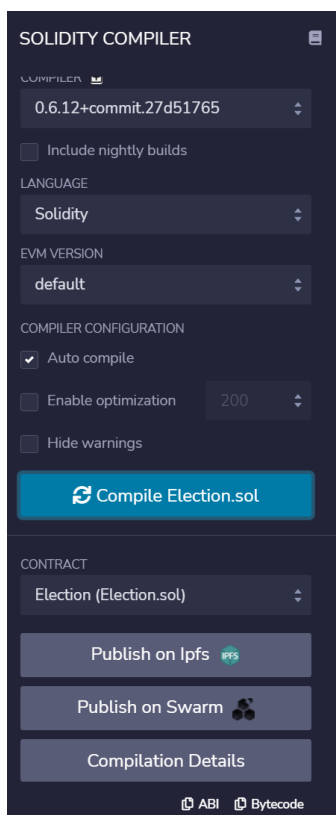
j) J'ajoute l'ensemble des fichiers sol sur mon environnement Remix.



k) Je vais ensuite compiler mon smart contract « Election » et je vais ensuite récupérer et enregistrer l'ABI ainsi que le Byte code du contrat dans un fichier txt.



On attend que ca compile.



Une fois que c'est compilé, j'ai copié et enregistré les fichiers ABI.txt et Bytecode.txt.

l) Je clique ensuite sur deploy puis je remonte la fenêtre , on a les détails de la transaction.

The screenshot shows the 'DEPLOY & RUN TRANSACTIONS' window in the Remix IDE. On the left, the 'ENVIRONMENT' is set to 'JavaScript VM (London)', the 'ACCOUNT' is '0x5B3...eddC4 (99.9999999%)', and the 'GAS LIMIT' is '3000000'. The 'VALUE' is '0 Wei'. The 'CONTRACT' is 'Election - contracts/Election.sol'. The 'Deploy' button is highlighted. Below it, there are options to 'Publish to IPFS' or 'At Address'. The 'Transactions recorded' section shows '1' transaction. The 'Deployed Contracts' section shows 'ELECTION AT 0xD91...39138 (MEMOR)'. On the right, the 'StructDefinition Candidate' section shows '3 reference(s)'. The 'Search with transaction hash or address' field is empty. The 'Debug' button is visible. The main area displays the transaction details for the 'Election' contract constructor.

```
2
3 // SPDX-License-Identifier: GPL-3.0
4
5 import "./Ownable.sol";
6 import "./SafeMath.sol";
7
8 contract Election is Ownable {
```

StructDefinition Candidate 3 reference(s)

Search with transaction hash or address

[vm] from: 0x5B3...eddC4 to: Election.(constructor) value: 0 wei data: 0x608...c0033 logs: 0
hash: 0x9b2...d651e

status true Transaction mined and execution succeed

transaction hash 0x9b28257e1feb4734103bf15026f153415c31882e6e32e029dbe9dee4e47d651e

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Election.(constructor)

gas 8000000 gas

transaction cost 554506 gas

execution cost 554506 gas

input 0x608...c0033

decoded input {}

decoded output -

logs []

val 0 wei

m) Les frais de transaction (transaction cost) ne sont pas identiques à ceux de l'énoncé du TP car le prix du gas est différent.

L'adresse publique de mon smart contract est : 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

n et o) Je vais déployer mon smart en ajoutant mon nom de famille.

The screenshot shows the 'DEPLOY & RUN TRANSACTIONS' window in the Remix IDE. The 'Publish to IPFS' checkbox is checked. The 'At Address' button is highlighted. The 'Transactions recorded' section shows '1' transaction. The 'Deployed Contracts' section shows 'ELECTION AT 0xD91...39138 (MEMOI)'. The 'addCandidate' function is selected, and the 'KASDI' value is entered. The 'transferOwner...' function is also visible, with the 'address newOwner' value entered. The 'vote' function is also visible, with the 'uint256_candidateId' value entered. The 'candidates' function is also visible, with the 'uint256' value entered. The 'candidatesCou...' function is also visible. The 'owner' function is also visible. The 'voters' function is also visible, with the 'address' value entered. The 'Low level interactions' section is visible, with the 'CALLDATA' field and the 'Transact' button.

DEPLOY & RUN TRANSACTIONS

Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 1

Deployed Contracts

ELECTION AT 0xD91...39138 (MEMOI)

addCandidate KASDI

transferOwner... address newOwner

vote uint256_candidateId

candidates uint256

candidatesCou...

owner

voters address

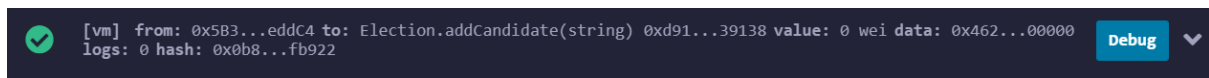
Low level interactions

CALLDATA

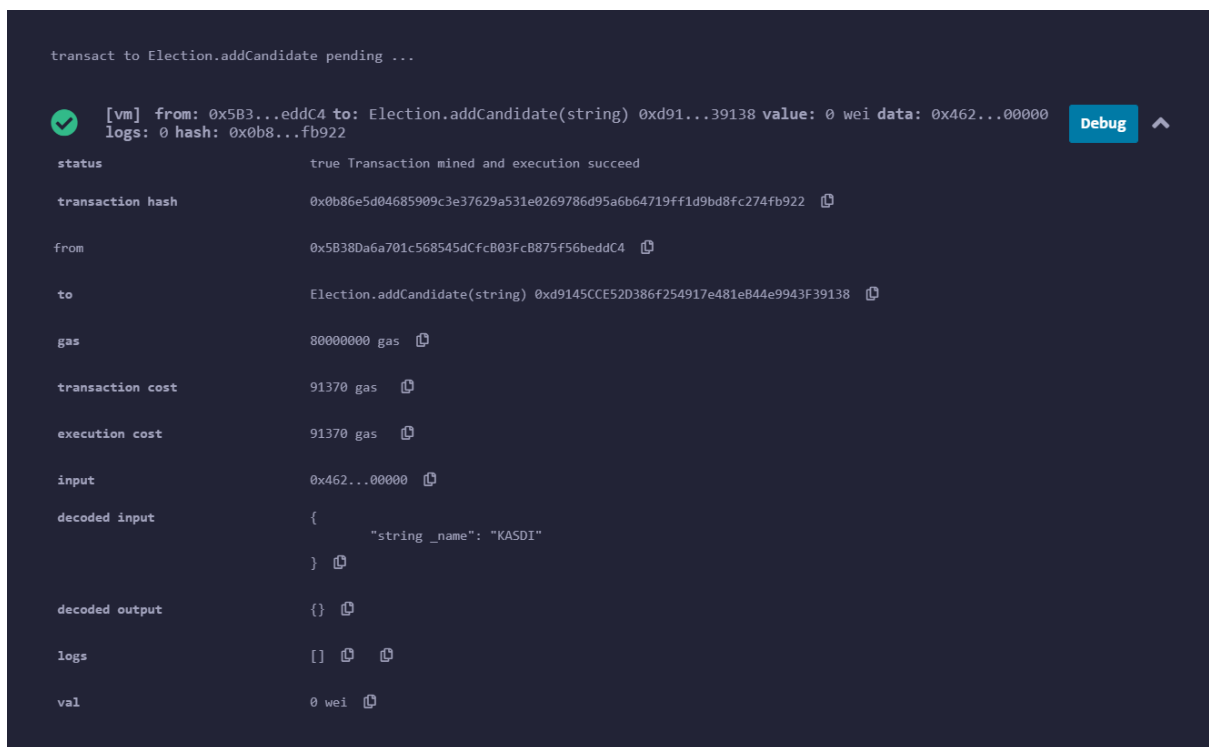
Transact

Pour cela, j'ajoute mon nom de famille dans « addCandidate » puis je clique dessus

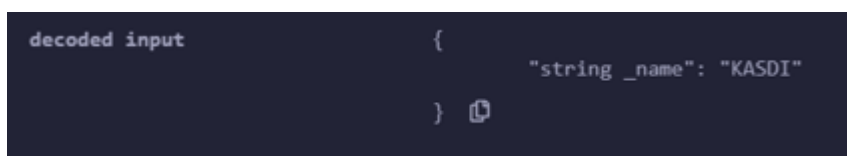
On remarque qu'une nouvelle transaction s'est créée



Dont voici les détails :



p) Voici le candidate ID :



q) J'ai fait quelques transactions test entre temps donc je rajoute maintenant mon quatrième candidat en ajoutant le nom « HOSSAIN »

ELECTION AT 0XD91...39138 (MEMOI)

addCandidate

HOSSAIN

transferOwner...

address newOwner

vote

uint256 _candidateId

candidates

uint256

candidatesCou...

owner

voters

address

Low level interactions

CALLDATA

Transact

On remarque qu'une nouvelle transaction s'est créée :

[vm] from: 0x5B3...eddC4 to: Election.addCandidate(string) 0xd91...39138 value: 0 wei data: 0x462...00000
logs: 0 hash: 0xb17...cca2f

Debug

Dont voici les détails :

[vm] from: 0x5B3...eddC4 to: Election.addCandidate(string) 0xd91...39138 value: 0 wei data: 0x462...00000
logs: 0 hash: 0xb17...cca2f

Debug

status

true Transaction mined and execution succeed

transaction hash

0xb178d0aa3d70c528e0dcffd40dce26e9206634b087dc4ebf8882d073826cca2f

from

0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to

Election.addCandidate(string) 0xd9145CCE52D386f254917e481eB44e9943F39138

gas

80000000 gas

transaction cost

74294 gas

execution cost

74294 gas

input

0x462...00000

decoded input

{
"string _name": "HOSSAIN"
}

decoded output

{}

logs

[]

val

0 wei

r) Voici la valeur du quatrième candidate ID :

decoded input

{
"string _name": "HOSSAIN"
}

s) Pour trouver l'adresse du propriétaire, il faut cliquer sur « Owner ».

addCandidate HOSSAIN ▼

transferOwner... address newOwner ▼

vote uint256 _candidateId ▼

candidates uint256 ▼

candidatesCou...

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

voters address ▼

Low level interactions ⓘ

CALLDATA

Transact

On constate qu'une transaction apparait :

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

from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

to Election.owner() 0xd9145CCE52D386f254917e481eB44e9943F39138

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

input 0x8da...5cb5b

decoded input {}



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

logs []

Voici l'adresse du propriétaire :

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

t) Afin de réaliser le vote, je rajoute l'ID du candidat à côté de « Candidate » puis je clique sur « vote »

▼ ELECTION AT 0XD91...39138 (MEMO)  

addCandidate

HOSSAIN

▼

transferOwner...

address newOwner

▼

vote

1

▼

candidates

1

▼

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

candidatesCou...

0: uint256: 3


owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

voters

address

▼



[vm] from: 0x5B3...eddC4 to: Election.vote(uint256) 0xd91...39138 value: 0 wei data: 0x012...00001 logs: 1


Debug ^

hash: 0xf2f...84da1


status

true Transaction mined and execution succeed


transaction hash

0xf2f4d8ad3d4b7757994095b2d55ebb474780c1e505209d2546301c95ea384da1 


from

0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 


to

Election.vote(uint256) 0xd9145CCF52D386f254917e481eB44e9943F39138 


gas

8000000 gas 


transaction cost

69467 gas 

execution cost


69467 gas 

input

0x012...00001 


decoded input

{
 "uint256_candidateId": "1"
}



decoded output

{}



```
logs      [      {      {      "from": "0xd9145CCE52D386f254917e481e844e9943f39138",      "topic": "0xffff3c900d938d21d0990d786e819f29b8d05c1ef587b462b939609625b684b16",      "event": "votedEvent",      "args": {      "0": "1",      "_candidateId": "1"      }      }      }      ]      val      0 wei      call to Election.candidates
```

u) On remarque que voteCount s'est incremented à 1, ce qui montre que mon premier candidat a voté.

▼ ELECTION AT 0XD91...39138 (MEMO)

addCandidate

HOSSAIN

transferOwner...

address newOwner

vote

1

candidates

1

0: uint256: id 1

1: string: name KASDI

2: uint256: voteCount 1

candidatesCou...

0: uint256: 3

owner

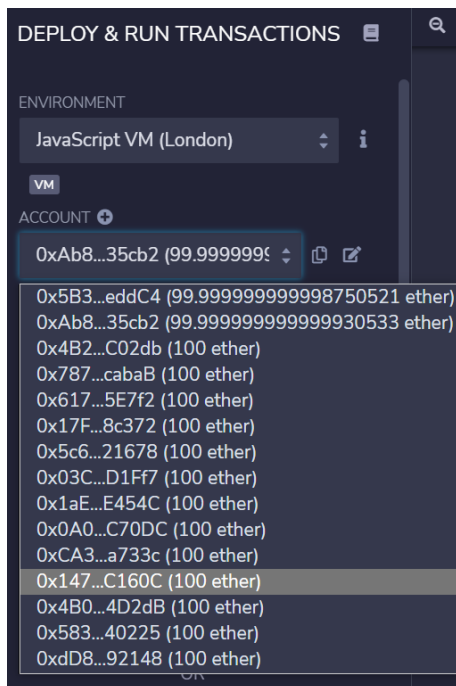
0: address: 0x5B38Da6a701c568545dCfcB03F
cB875f56beddC4

voters

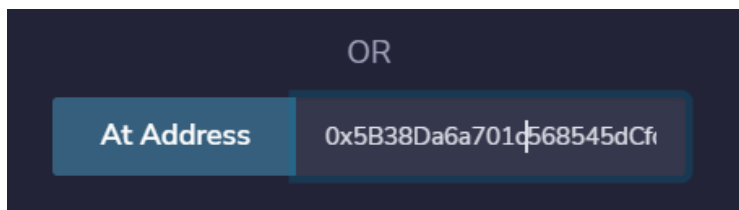
address

```
decoded output      {      "0": "uint256: id 1",      "1": "string: name KASDI",      "2": "uint256: voteCount 1"
```


v) On change de compte en choisissant un autre dans Account



On ajoute l'adresse publique de notre smart contract



Et en ajoutant un quatrième candidat , on peut voter

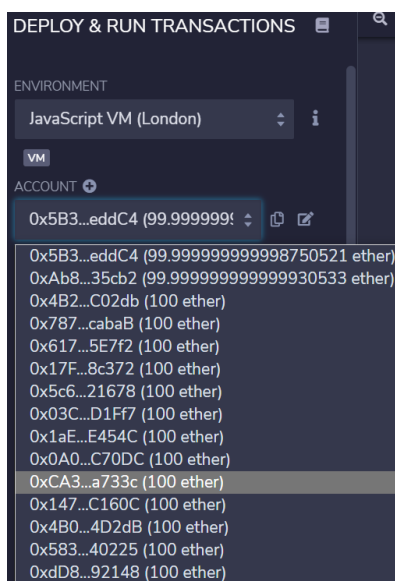


```
logs      [
          {
            "from": "0xd9145CCe52D386f254917e481e844e9943f39138",
            "topic": "0xffff3c900d938d21d0990d786e819f29b8d05c1ef587b462b939609625b684b16",
            "event": "votedEvent",
            "args": {
              "0": "4",
              "_candidateId": "4"
            }
          }
        ]  [D] [D]

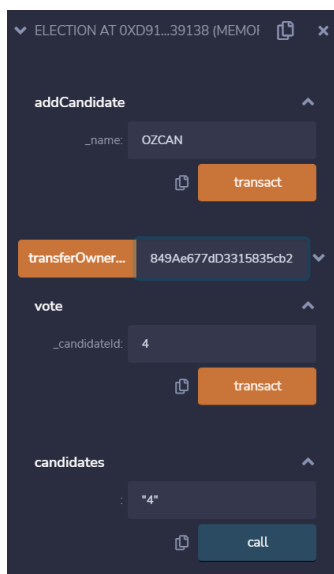
val      0 wei  [D]
```

On peut donc voir que le quatrième candidat a voté.

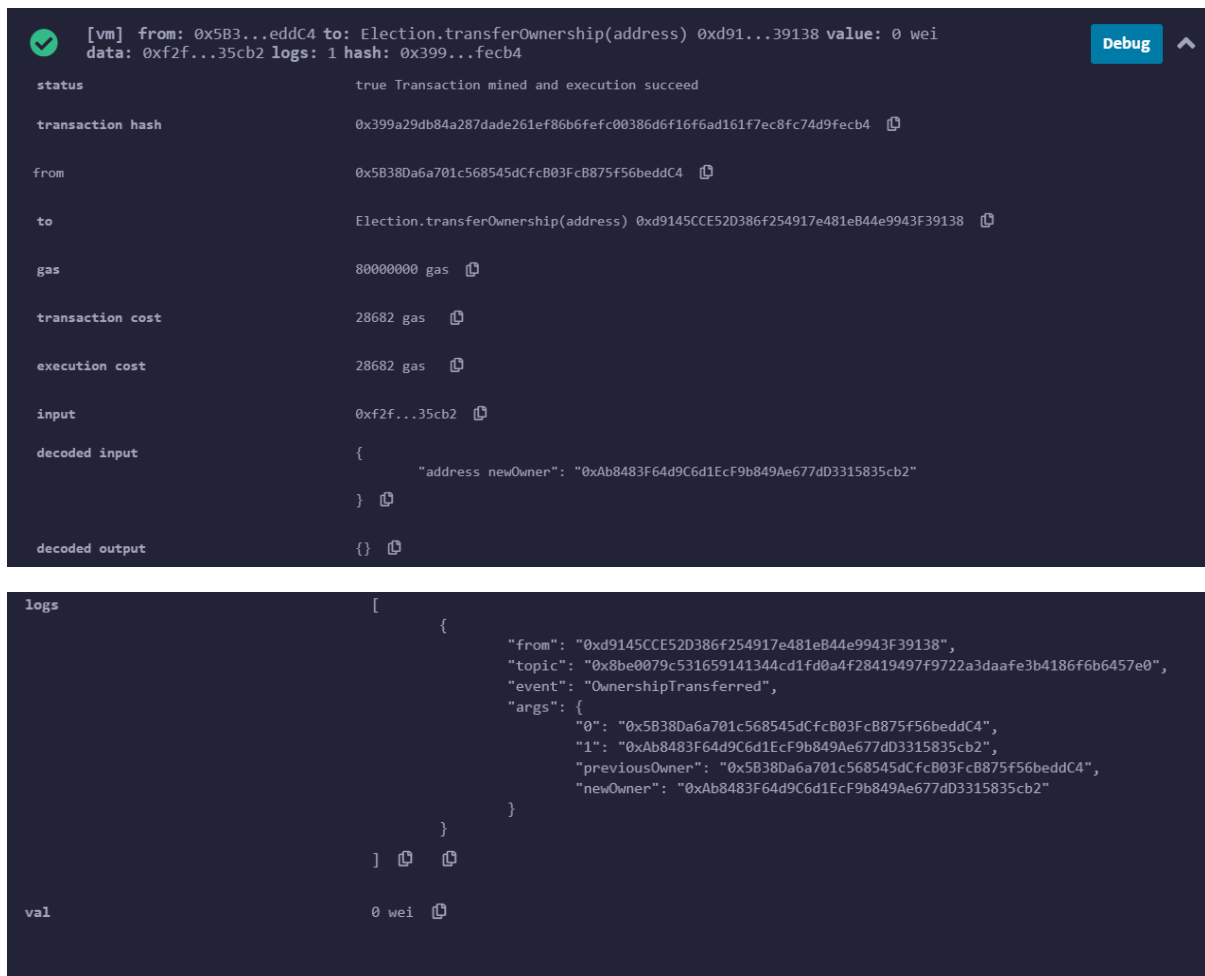
w) On remet le propriétaire de base :



Je rajoute ensuite l'adresse publique du second compte :



On clique sur « transferOwnership », le transfert de propriété a ainsi été fait.



The screenshot displays a transaction interface with a green checkmark icon and a 'Debug' button. The transaction details are as follows:

- status:** true Transaction mined and execution succeed
- transaction hash:** 0x399a29db84a287dade261ef86b6f6fc00386d6f16f6ad161f7ec8fc74d9fecb4
- from:** 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
- to:** Election.transferOwnership(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
- gas:** 80000000 gas
- transaction cost:** 28682 gas
- execution cost:** 28682 gas
- input:** 0xf2f...35cb2
- decoded input:** { "address newOwner": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2" }
- decoded output:** {}

The logs section shows a single log entry with the following details:

- from:** "0xd9145CCE52D386f254917e481eB44e9943F39138",
- topic:** "0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0",
- event:** "OwnershipTransferred",
- args:** {
 - 0:** "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
 - 1:** "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
 - previousOwner:** "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
 - newOwner:** "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"}

The value section shows the result: 0 wei.

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) Je vais ensuite modifier le code afin de faire en sorte que je sois la seule 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");
}
```

