

TIC-BLK4

Etape 2 - Partie 2


“Code and Deploy a smart contract on Ethereum testnet Ropsten”


Thomas SIMOES

Demande 1 etherium from faucet:


# MetaMask Ether Faucet


Nombre d'ethererium sur le réseau ropsten:

 **METAMASK**

**Account 1**

Détails


0x7b66...A994 

 **18 ETH**

Vous ne voyez pas vos jetons ?









Cliquez sur Ajouter un jeton pour les ajouter à votre compte

[Ajouter un jeton](#)

 **18 ETH**

DéposerEnvoyer

Historique

 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		
 Déposer	CONFIRMÉ	1 ETH
10/6/2019 at 15:38		

## Compilation du smart contract:

The screenshot shows the Solidity Compiler interface in the Remix IDE. On the left, the 'Compiler' section is set to '0.5.12+commit.7709ce', 'Language' is 'Solidity', and 'EVM Version' is 'compiler default'. The 'Compiler Configuration' section has 'Auto compile' disabled and 'Enable optimization' checked. The 'Contract' dropdown shows 'Election (election.sol)'. Below this, there are buttons for 'Publish on Swarm', 'Publish on Ipfs', and 'Compilation Details'. The 'ABI' and 'Bytecode' tabs are visible. The main editor displays the Solidity code for the 'Election' contract, which includes a 'Candidate' struct, a 'voters' mapping, a 'candidates' mapping, and functions for adding candidates and voting. The bottom status bar shows 'ContractDefinition Election' and '0 reference(s)'.

```
1 pragma solidity ^0.5.0;
2
3 contract Election {
4     // Model a Candidate
5     struct Candidate {
6         uint id;
7         string name;
8         uint voteCount;
9     }
10
11     // Store accounts that have voted
12     mapping(address => bool) public voters;
13     // Store Candidates
14     // Fetch Candidate
15     mapping(uint => Candidate) public candidates;
16     // Store Candidates Count
17     uint public candidatesCount;
18
19     // voted event
20     event votedEvent (
21         uint indexed _candidateId
22     );
23
24     constructor() public {
25         addCandidate("Candidate 1");
26         addCandidate("Candidate 2");
27     }
28
29     function addCandidate (string memory _name) public {
30         candidatesCount ++;
31         candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
32     }
33
34     function vote (uint _candidateId) public {
35         // require that they haven't voted before
36         require(!voters[msg.sender]);
37
38         // require a valid candidate
39         require(_candidateId > 0 && _candidateId <= candidatesCount);
40
41         // record that voter has voted
42         voters[msg.sender] = true;
43
44         // update candidate vote Count
45         candidates[_candidateId].voteCount ++;
46
47         // trigger voted event
```

## Injection Web3:

The screenshot shows the Solidity Compiler interface in the Remix IDE, now in the 'DEPLOY & RUN TRANSACTIONS' view. The 'Environment' is set to 'Injected Web3'. The 'Account' dropdown shows '0x7b6...ca994 (18 ether)'. The 'Gas limit' is set to '3000000' and the 'Value' is '0 wei'. The 'Contract' dropdown shows 'Election - browser/election.sol'. The 'Deploy' button is highlighted. Below this, there are buttons for 'At Address' and 'Load contract from Address'. The 'Transactions recorded' section shows '0' transactions. The 'Deployed Contracts' section shows 'Currently you have no contract instances to interact with.' The main editor displays the same Solidity code for the 'Election' contract as in the previous screenshot. The bottom status bar shows 'ContractDefinition Election' and '0 reference(s)'.

```
1 pragma solidity ^0.5.0;
2
3 contract Election {
4     // Model a Candidate
5     struct Candidate {
6         uint id;
7         string name;
8         uint voteCount;
9     }
10
11     // Store accounts that have voted
12     mapping(address => bool) public voters;
13     // Store Candidates
14     // Fetch Candidate
15     mapping(uint => Candidate) public candidates;
16     // Store Candidates Count
17     uint public candidatesCount;
18
19     // voted event
20     event votedEvent (
21         uint indexed _candidateId
22     );
23
24     constructor() public {
25         addCandidate("Candidate 1");
26         addCandidate("Candidate 2");
27     }
28
29     function addCandidate (string memory _name) public {
30         candidatesCount ++;
31         candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
32     }
33
34     function vote (uint _candidateId) public {
35         // require that they haven't voted before
36         require(!voters[msg.sender]);
37
38         // require a valid candidate
39         require(_candidateId > 0 && _candidateId <= candidatesCount);
40
41         // record that voter has voted
42         voters[msg.sender] = true;
43
44         // update candidate vote Count
45         candidates[_candidateId].voteCount ++;
46
47         // trigger voted event
```

## Déploiement sur le réseau ropsten:

The screenshot shows a web interface for deploying a smart contract on the Ropsten test network. The interface includes a gas fee of 0.000422 ETH and a total of 0.000422 ETH. It also features buttons for 'Rejeter' and 'Confirmer'. On the right, the Solidity code for the 'election.sol' contract is displayed, showing the constructor, addCandidate, and vote functions.

```
// election.sol
// Model a Laminate
struct Candidate {
    uint id;
    string name;
    uint voteCount;
}

// Store accounts that have voted
mapping(address => bool) public voters;
// Store Candidates
// Fetch Candidate
mapping(uint => Candidate) public candidates;
// Store Candidates Count
uint public candidatesCount;

// voted event
event votedEvent (
    uint indexed _candidateId
);

constructor() public {
    addCandidate("Candidate 1");
    addCandidate("Candidate 2");
}

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

function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);

    // require a valid candidate
    require(_candidateId > 0 && _candidateId <= candidatesCount);

    // record that voter has voted
    voters[msg.sender] = true;

    // update candidate vote Count
    candidates[_candidateId].voteCount ++;

    // trigger voted event
    emit votedEvent(_candidateId);
}
```

## Interaction avec le smart contract déployé sur ropsten:

The screenshot shows a web interface for interacting with a smart contract on the Ropsten test network. The interface displays the 'Election - browser/election.sol' contract and its functions. It also shows the 'Deployed Contracts' section with the contract address 0x0bA...2a7AD. The 'Interact' section shows the 'addCandidate' function being called with the argument 'Candidate 2'.

```
// election.sol
// Model a Laminate
struct Candidate {
    uint id;
    string name;
    uint voteCount;
}

// Store accounts that have voted
mapping(address => bool) public voters;
// Store Candidates
// Fetch Candidate
mapping(uint => Candidate) public candidates;
// Store Candidates Count
uint public candidatesCount;

// voted event
event votedEvent (
    uint indexed _candidateId
);

constructor() public {
    addCandidate("Candidate 1");
    addCandidate("Candidate 2");
}

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

function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);

    // require a valid candidate
    require(_candidateId > 0 && _candidateId <= candidatesCount);

    // record that voter has voted
    voters[msg.sender] = true;

    // update candidate vote Count
    candidates[_candidateId].voteCount ++;

    // trigger voted event
    emit votedEvent(_candidateId);
}
```

Liste des transactions sur etherscan:

Etherscan

Ropsten Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens

Q

Home

Blockchain

Tokens

Misc

Ropsten

Address

0x7b66f769198f25341d4AEf5E349Dc550422cA994

Overview

Balance: 17.999089222 Ether

More Info

My Name Tag: Not Available

Transactions

1/2 Latest 21 txns

Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0xa12d2b09b31628...	6509323	1 min ago	0x7b66f769198f253...	OUT 0xbcb638831eb089...	0 Ether	0.000066064
0x411baef40ee415...	6509310	6 mins ago	0x7b66f769198f253...	OUT Contract Creation	0 Ether	0.000423357
0xad9e086a53390...	6509302	9 mins ago	0x7b66f769198f253...	OUT Contract Creation	0 Ether	0.000423357
0xf0bc20dbf669cb5...	6509286	13 mins ago	0x81b7e08f65bdf56...	IN 0x7b66f769198f253...	1 Ether	0.000021
0x5ac833dba09aac...	6509286	13 mins ago	0x81b7e08f65bdf56...	IN 0x7b66f769198f253...	1 Ether	0.000021
0xd35c0bea27b3d...	6509286	13 mins ago	0x81b7e08f65bdf56...	IN 0x7b66f769198f253...	1 Ether	0.000021
0xe7a1f92679b99...	6509286	13 mins ago	0x81b7e08f65bdf56...	IN 0x7b66f769198f253...	1 Ether	0.000021
0x4007215016642d...	6509286	13 mins ago	0x81b7e08f65bdf56...	IN 0x7b66f769198f253...	1 Ether	0.000021

Transaction du vote:

Etherscan

Ropsten Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Ens

Q

Home

Blockchain

Tokens

Misc

Ropsten

Transaction Details

Overview

Event Logs (1)

State Changes

[ This is a Ropsten Testnet transaction only ]

Transaction Hash:

0xa12d2b09b316286911fcbca2e3534d2d85e2d19cfd29ebe5bdf535ce73119bfa

Status:

Success

Block:

6509323 3 Block Confirmations

Timestamp:

2 mins ago (Oct-06-2019 01:50:59 PM +UTC)

From:

0x7b66f769198f25341d4AEf5E349Dc550422cA994

To:

Contract 0xbcb638831eb08969512c986b5d1aafcd89e6a192

Value:

0 Ether (\$0.00)

Transaction Fee:

0.000066064 Ether (\$0.000000)

Click to see More

Vote réalisé:

candidates

1

▼

0: uint256: id 1

1: string: name Candidate 1

2: uint256: voteCount 1