

How to Write and Deploy a Smart Contract

Anastasia Lalamentik

Full Stack Engineer @ Kaleido

October 16, 2023



**OCTOBER
15•16•17**

Raleigh, NC USA



Table of Contents

- Tools needed
- How smart contracts work
- Quick overview of Solidity
- Build smart contract together
- Deploy smart contract on an Ethereum test network
- Interact with smart contract



OCTOBER 15 - 17 • RALEIGH, NC USA

Tools needed


- Set up a Node development environment on your machine
 - You should be able to run **node** and **npm** commands from your Terminal
 - **Hardhat** - `npm install --save-dev hardhat`
 - **MetaMask Wallet**: Make sure you have some Sepolia Testnet ETH in your wallet. Use a Sepolia Faucet to fund your account.
 - **Infura**: Sign up for a free account and create a Sepolia network endpoint
- Full details in README found in <https://github.com/lanasta/deploy-smart-contract/tree/ato2023>

Show test networks

Select this to show test networks in network list



Sepolia PoW Faucet

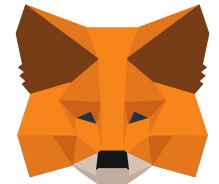


Target Address:
0x5Fe236Ac2A50ba38ba6F091c7431f07c08F13bc6

Your Mining Reward: 0.023 SepETH Current Hashrate: 783.8 H/s

Number of Workers:	10 / 10	+ -
Minimum Claim Reward:	0.05 SepETH	
Maximum Claim Reward:	2.5 SepETH	
Remaining Session Time:	11h 58min	
Total Shares:	14	
Avg. Reward per Hour:	1.27 SepETH/h	
Reward Boost:	+ 0%	Boost

Stop Mining



OCTOBER 15 - 17 • RALEIGH, NC USA

How smart contracts work

- Runs in a blockchain network: decentralized, trustless, transparent, immutable
- Self-execute when certain conditions are met, removes the need for a third-party (eg. centralized authorities like banks and intermediaries)
 - e.g. An artist no longer has to depend on record labels or distribution companies to receive royalties when someone buys their music. Percentage of shares coded in smart contract, cannot be tampered with.
- Widely used in business management, legal processes, financial agreements, health systems
- In enterprise use cases, a private blockchain would be of interest



Smart contract concepts

- **Ethereum Virtual Machine(EVM)**: the execution environment for smart contracts in Ethereum. “World computer”, accessible to anyone around the world through participating Ethereum nodes. Each node runs a copy of the EVM.
- **Gas fees**: required to compensate miners who use their computing power (usually referred to as running nodes in blockchain) to validate smart contracts, process and confirm transactions – variable based on traffic
- **Block explorer**: A visualization tool that allows anyone to explore the state of a particular blockchain network and see information about blocks, transactions, balances, etc e.g. <https://etherscan.io/>
- **Addresses**: Each smart contract and each Ethereum account has an address that is represented by a 42-character hexadecimal address e.g. `0x92bc44d5318309EE2abF1539BF71dE1b7d7bE3b8`
- **Application Binary Interface(ABI)**: defines the methods and variables that are available to interact with in a smart contract



THE EVM

For the **EVM** to be able to run your contract it needs to be in **bytecode**. Compilation turns this:

```
1 pragma solidity 0.4.24;
2
3 contract Greeter {
4
5     function greet() public constant returns (string) {
6         return "Hello";
7     }
8
9 }
```

into this

```
1 PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0x41  
JUMPI PUSH1 0x0 CALLDATALOAD PUSH29  
0x1000000000000000000000000000000000000000000000000000000000000000 SWAP1  
DIV PUSH4 0xFFFFFFFF AND DUP1 PUSH4 0xCFAE3217 EQ PUSH2 0x46 JUMPI  
JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO PUSH2  
0x52 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0x5B PUSH2 0xD6  
JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 DUP1 PUSH1 0x20 ADD DUP3 DUP2  
SUB DUP3 MSTORE DUP4 DUP2 DUP2 MLOAD DUP2 MSTORE PUSH1 0x20 ADD  
SWAP2 POP DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4 DUP4 PUSH1  
0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x9B JUMPI DUP1 DUP3 ADD
```

What bytecode looks like

Source:
<https://ethereum.org/en/developers/docs/smart-contracts/compiling/>

```
1  [
2      {
3          "constant": true,
4          "inputs": [],
5          "name": "greet",
6          "outputs": [
7              {
8                  "name": "",
9                  "type": "string"
10             }
11         ],
12         "payable": false,
13         "type": "function"
14     }
15 ]
```

Corresponding ABI for the Solidity code



OCTOBER 15 - 17 • RALEIGH, NC USA

Gas fees

Sub-units of Ether

Unit	wei value	wei	ether value
wei	1 wei	1	10 ⁻¹⁸ ETH
kwei	10 ³ wei	1,000	10 ⁻¹⁵ ETH
mwei	10 ⁶ wei	1,000,000	10 ⁻¹² ETH
gwei	10 ⁹ wei	1,000,000,000	10 ⁻⁹ ETH
microether	10 ¹² wei	1,000,000,000,000	10 ⁻⁶ ETH
milliether	10 ¹⁵ wei	1,000,000,000,000,000	10 ⁻³ ETH
ether	10 ¹⁸ wei	1,000,000,000,000,000,000	1 ETH

Source: Ethersc

Transaction Details < >

Sponsored: OpenOcean: The most efficient aggregator on Ethereum & 16 other networks. [Swap your crypto](#) now!

OverviewStateComments

Transaction Hash:

0xe2333f02a9dde721c1163a3c170cc7e7a1f58341e2e6403277365426f486a0a5

Status:

Success

Block:

15440635 1 Block Confirmation

Timestamp:

54 secs ago (Aug-30-2022 01:34:59 PM +UTC) | Confirmed within 30 secs

From:

0xea674fdde714fd979de3edf0f56aa9716b898ec8 (Ethereum)

To:

0x9dca330f5469374950dc23664ab10ee6a7fdd06c

Value:

0.099465521973971281 Ether (\$157.18)

Transaction Fee:

0.000545644077951 Ether (\$0.86)

Gas Price:

0.000000025983051331 Ether (25.983051331 Gwei)

Gas Limit & Usage by Txn:

250,000 | 21,000 (8.4%)

Gas Fees:

Base: 24.983051331 Gwei | Max: 50.715594201 Gwei | Max Priority: 1 Gwei

Burnt & Txn Savings Fees:

Burnt: 0.000524644077951 Ether (\$0.83)

Txn Savings: 0.00051938340027 Ether (\$0.82)

Source: Etherscan.io transaction details



OCTOBER 15 - 17 • RALEIGH, NC USA

Solidity Overview

1. **Version pragma:** solidity compiler version compatible with the code at the time of the development. Future versions may contain incompatible changes.
2. **Contract keyword:** the contract name follows after, encapsulates the smart contract logic
3. **State variables:** types include signed integers, unsigned integers, Boolean, addresses, enums, and bytes
4. **Function declarations:** can take in parameter, specify view or pure functions, return type and function visibility types – private, internal, external, or public.

Source: <https://www.geeksforgeeks.org/introduction-to-solidity/>

```
pragma solidity >=0.4.0 <0.6.0;
contract SimpleStorage {
    uint storedData;
    function set(uint x) public {
        storedData = x;
    }
    function get() public view returns (uint) {
        return storedData;
    }
}
```

Source: https://www.tutorialspoint.com/solidity/solidity_quick_guide.htm



OCTOBER 15 - 17 • RALEIGH, NC USA

Solidity Global Variables

Name	Returns
blockhash(uint blockNumber) returns (bytes32)	Hash of the given block - only works for 256 most recent, excluding current, blocks
block.coinbase (address payable)	Current block miner's address
block.difficulty (uint)	Current block difficulty
block.gaslimit (uint)	Current block gaslimit
block.number (uint)	Current block number
block.timestamp (uint)	Current block timestamp as seconds since unix epoch
gasleft() returns (uint256)	Remaining gas
msg.data (bytes calldata)	Complete calldata
msg.sender (address payable)	Sender of the message (current caller)
msg.sig (bytes4)	First four bytes of the calldata (function identifier)
msg.value (uint)	Number of wei sent with the message
now (uint)	Current block timestamp
tx.gasprice (uint)	Gas price of the transaction
tx.origin (address payable)	Sender of the transaction

Source: https://www.tutorialspoint.com/solidity/solidity_variables.htm



OCTOBER 15 - 17 • RALEIGH, NC USA

Other Solidity Features

1. **struct**: describes an object containing fields of different types
2. **modifier**: used to check pre-conditions for functions
3. **event**: logging capability enabling smart contracts to notify external applications of new events and changes on the blockchain

```
struct WorkshopAttendee {  
    string name;  
    string city;  
    bool firstTimeAtAto;  
}
```

```
contract HouseOwnership {  
  
    address public owner;  
    uint256 public price;  
  
    constructor() {  
        owner = msg.sender;  
    }  
  
    event OwnershipChanged(address oldOwner, address newOwner);  
  
    modifier onlyOwner() {  
        require(msg.sender == owner, "You are not the owner!");  
        _;  
    }  
  
    function changeOwner(address newOwner) public onlyOwner {  
        address oldOwner = owner;  
        owner = newOwner;  
        emit OwnershipChanged(oldOwner, newOwner);  
    }  
  
    function setPrice(uint256 _price) public onlyOwner {  
        price = _price;  
    }  
}
```



**Let's write and deploy a
smart contract together!**



OCTOBER 15 - 17 • RALEIGH, NC USA

Thank You



[/lanasta/deploy-smart-contract](#)



[/in/anastasiyalamentik](#)



anastasia.lalamentik@kaleido.io



OCTOBER 15 - 17 • RALEIGH, NC USA