

## KASEICOIN.SOL COMPILED

The screenshot displays the Solidity Compiler web interface. On the left, the 'SOLIDITY COMPILER' sidebar contains settings for the compiler (0.5.17+commit.d19bba13), language (Solidity), and compiler configuration (Auto compile, Enable optimization, Hide warnings). A 'Compile KaseiCoin.sol' button is visible. Below the compiler settings, the 'CONTRACT' section shows 'KaseiCoin (KaseiCoin.sol)' with options to 'Publish on Ipfs' and view 'Compilation Details'. The main editor area shows the source code for 'KaseiCoin.sol', which includes imports from the OpenZeppelin library and a constructor for the KaseiCoin contract.

```
1 pragma solidity ^0.5.0;
2
3 // Import the following contracts from the OpenZeppelin library:
4 // * ERC20
5 // * ERC20Detailed
6 // * ERC20Mintable
7 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
8 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";
9 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Mintable.sol";
10
11 // Create a constructor for the KaseiCoin contract and have the contract inherit the libraries that you imported from OpenZeppelin.
12 contract KaseiCoin is ERC20, ERC20Detailed, ERC20Mintable {
13
14     constructor(
15         string memory name,
16         string memory symbol,
17         uint init_supply)
18         ERC20Detailed(name, symbol, 18)
19     {
20         public
21     }
22 }
23
24
25
26
```

## KASEICOINCROWDSALESOL COMPILED

The screenshot displays the Solidity Compiler interface with the file `KaseiCoinCrowdsale.sol` open. The left sidebar contains configuration options for the compiler, language (Solidity), and environment (compiler default). The main editor shows the Solidity code for the `KaseiCoinCrowdsale` contract, which inherits from `Crowdsale` and `MintedCrowdsale`. The code includes a constructor for the `KaseiCoinCrowdsale` contract and a `KaseiCoinCrowdsaleDeployer` contract that creates instances of the `KaseiCoin` and `KaseiCoinCrowdsale` contracts. The bottom panel shows the compilation status, indicating that the contract was compiled successfully.

```
1 pragma solidity ^0.5.0;
2
3 import "../KaseiCoin.sol";
4 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";
6
7
8 // Make the KaseiCoinCrowdsale contract inherit the following OpenZeppelin:
9 // * Crowdsale
10 // * MintedCrowdsale
11 contract KaseiCoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CONTRACT SIGNATURE TO ADD INHERITANCE
12
13     // Provide parameters for all of the features of your crowdsale, such as the 'rate', 'wallet' for fundraising, and 'token'.
14     constructor(uint rate, address payable wallet, KaseiCoin token) Crowdsale(rate, wallet, token) public {
15     }
16 }
17
18
19 contract KaseiCoinCrowdsaleDeployer {
20     // Create an 'address public' variable called 'kasei_token_address'.
21     address public kasei_token_address;
22     // Create an 'address public' variable called 'kasei_crowdsale_address'.
23     address public kasei_crowdsale_address;
24
25
26     // Add the constructor.
27     constructor(string memory name, string memory symbol, address payable wallet) public {
28         // Create a new instance of the KaseiCoin contract.
29         KaseiCoin token = new KaseiCoin(name, symbol, 0);
30
31         // Assign the token contract's address to the 'kasei_token_address' variable.
32         kasei_token_address = address(token);
33
34
35         // Create a new instance of the 'KaseiCoinCrowdsale' contract
36         KaseiCoinCrowdsale crowdsale = new KaseiCoinCrowdsale(1, wallet, token);
37
38         // Assign the 'KaseiCoinCrowdsale' contract's address to the 'kasei_crowdsale_address' variable.
39         kasei_crowdsale_address = address(crowdsale);
40     }
41 }
```

CONTRACT: KaseiCoinCrowdsale (KaseiCoinCrowdsale)

Publish on Ethers

Compilation Details

0 0 System on network Search with transaction hash or address

The following libraries are accessible:

- `openzeppelin` version 1.0.1
- `ethers` 1.0.1
- `web3` from `web3.js` for more info

## KASEICOINCROWDSALE.SOL DEPLOYER COMPILED

The screenshot displays the Solidity Compiler interface with the following components:

- SOLIDITY COMPILER** sidebar on the left:
  - COMPILER**: 0.5.17+commit.d19bba13
  - LANGUAGE**: Solidity
  - ENV VERSION**: compiler default
  - COMPILER CONFIGURATION**:
    - ☐ Include nightly builds
    - ☐ Auto compile
    - ☐ Enable optimization (200)
    - ☐ Hide warnings
  - Compile** button: KaseiCoinCrowdsale.sol
  - CONTRACT**: KaseiCoinCrowdsale (KaseiCoinCrowdSale)
  - Buttons**: Publish on Ipfs, Compilation Details
  - Links**: ABI, Bytecode
- Main Editor**:
  - File explorer at the top shows: Home, KaseiCoin.sol, KaseiCoinCrowdsale.sol
  - Line numbers 1-30 are visible on the left of the code.
  - Solidity code for `KaseiCoinCrowdsale.sol` is displayed, including imports, comments, and contract definitions for `KaseiCoinCrowdsale` and `KaseiCoinCrowdsaleDeployer`.
- Bottom Section**:
  - Network selection: "Select a network" with a dropdown arrow.
  - Search bar: "Search with transaction hash or address".
  - Footer note: "Right click on a JavaScript file in the file explorer and then click 'Run'".

## INITIAL DEPLOY

The screenshot displays the Remix IDE interface during the initial deployment of a KaseiCoin contract. The left sidebar, titled "DEPLOY & RUN TRANSACTIONS", shows the following configuration:

- ENVIRONMENT:** Injected Web3
- ACCOUNT:** 0xB31...6c5d1 (99.9695526)
- GAS LIMIT:** 3000000
- VALUE:** 0
- CONTRACT:** KaseiCoin - Kaseicoin.sol
- DEPLOY:** KaseiCoinCrowdSale (Symbol: ksc, Initial Supply: 100)
- Transactions recorded:** 1
- Deployed Contracts:** KASEICON AT 0x4C\_F0B2 (BLOCK)

The main editor displays the Solidity code for `KaseiCoinCrowdSale.sol`. The code includes a constructor for the `KaseiCoin` contract and a `KaseiCoinCrowdSaleDeployer` contract that creates a new instance of `KaseiCoinCrowdSale` and assigns it to the `kasei_token_address` variable.

```
14 constructor(uint rate, address payable wallet, KaseiCoin token) CrowdSale(rate, wallet, token) public {
15 }
16
17
18
19 contract KaseiCoinCrowdSaleDeployer {
20     // Create an 'address public' variable called 'kasei_token_address'.
21     address public kasei_token_address;
22     // Create an 'address public' variable called 'kasei_crowdsale_address'.
23     address public kasei_crowdsale_address;
24
25     // Add the constructor.
26     constructor(string memory name, string memory symbol, address payable wallet) public {
27         // Create a new instance of the KaseiCoin contract.
28         KaseiCoin token = new KaseiCoin(name, symbol, 0);
29
30         // Assign the token contract's address to the 'kasei_token_address' variable.
31         kasei_token_address = address(token);
32
33         // Create a new instance of the 'KaseiCoinCrowdSale' contract.
34         KaseiCoinCrowdSale crowdsale = new KaseiCoinCrowdSale(1, wallet, token);
35
36         // Assign the 'KaseiCoinCrowdSale' contract's address to the 'kasei_crowdsale_address' variable.
37         kasei_crowdsale_address = address(crowdsale);
38
39         // Set the 'KaseiCoinCrowdSale' contract as a minter.
40         token.addMinter(kasei_crowdsale_address);
41
42         // Have the 'KaseiCoinCrowdSaleDeployer' renounce its minter role.
43         token.renounceMinter();
44     }
45 }
46
47
48
49
50
```

The bottom panel shows the transaction log, indicating the successful creation of the KaseiCoin contract. The log entry is:

```
creation of KaseiCoin pending...
view on etherscan:
[Block:1 txIndex:0] from: 0xB31...6c5d1 to: KaseiCoin.constructor() value: 0 wei data: 0x00...000000 logs: 1 hash: 0x79a...12004
```

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM (London)

VM

ACCOUNT

0x5B3...edC4 (99.999999%)

GAS LIMIT

3000000

VALUE

0 Ether

CONTRACT

KaseiCoin - contracts/KaseiCoin.sol

DEPLOY

NAME

KaseiCoin

SYMBOL

ks

INT\_SUPPLY

100

INITIALIZE

Publish to IPFS

OR

At Address

Load contract from address

Transactions recorded

0

Deployed Contracts

KASEICOIN AT 0x0B1...3E13B (MEMOR)

KASEICOIN AT 0x0B1...33F4E (MEMOR)

KaseiCoin.sol

KaseiCoinCrowdsale.sol

```
9 // * Crowdsale
10 // * KaseiCoinCrowdsale
11 contract KaseiCoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CONTRACT SIGNATURE TO ADD INHERITANCE
12
13     // Provide parameters for all of the features of your crowdsale, such as the 'rate', 'wallet' for fundraising, and 'token'.
14     constructor(uint rate, address payable wallet, KaseiCoin token) Crowdsale(rate, wallet, token) public {
15     }
16 }
17
18
19 contract KaseiCoinCrowdsaleDeployer {
20     // Create an 'address public' variable called 'kasei_token_address'.
21     address public kasei_token_address;
22     // Create an 'address public' variable called 'kasei_crowdsale_address'.
23     address public kasei_crowdsale_address;
24
25
26     // Add the constructor.
27     constructor(string memory name, string memory symbol, address payable wallet) public {
28         // Create a new instance of the KaseiCoin contract.
29         KaseiCoin token = new KaseiCoin(name, symbol, 0);
30
31         // Assign the token contract's address to the 'kasei_token_address' variable.
32         kasei_token_address = address(token);
33
34
35         // Create a new instance of the 'KaseiCoinCrowdsale' contract.
36         KaseiCoinCrowdsale crowdsale = new KaseiCoinCrowdsale(1, wallet, token);
37
38         // Assign the 'KaseiCoinCrowdsale' contract's address to the 'kasei_crowdsale_address' variable.
39         kasei_crowdsale_address = address(crowdsale);
40
41         // Set the 'KaseiCoinCrowdsale' contract as a minter.
42         token.addMinter(kasei_crowdsale_address);
43
44         // Have the 'KaseiCoinCrowdsaleDeployer' renounce its minter role.
45         token.renounceMinter();
46     }
47 }
```

0

Listen on network

Search with transaction hash or address

[ve] from: 0x5B3...edC4 to: KaseiCoin.(constructor) value: 0 wei data: 0x0000...0000 logs: 1 hash: 0x1a...142781

creation of KaseiCoin pending...

[ve] from: 0x5B3...edC4 to: KaseiCoin.(constructor) value: 0 wei data: 0x0000...0000 logs: 1 hash: 0x4f6...1826fa