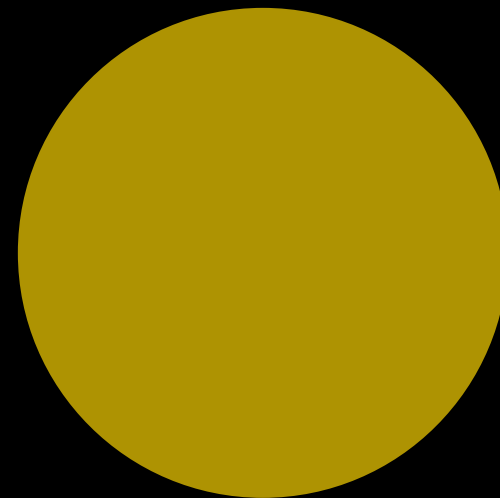


Crowdsale for:

# J3KOIN

BRIDGING ASSETS AND  
COMMUNITIES



Offered by J3K

# In this Presentation

Here's what we'll cover:

Who is J3K?

Our mission

Introduction of J3KOIN

Crowdsale

Client Experience

Polygon Network

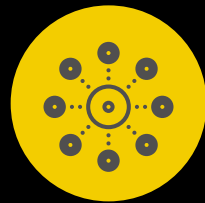
The Future

Q&A



# J3K

Making the inaccessible accessible



## DAO

Decentralized Autonomous Organization Designed to help ALL people establish and grow wealth



## Secure

Developed for the Blockchain to increase trust between parties, reduce costs/complexity, save time, and break down barriers



## Adaptable/Smart

Smart Contracts and Tokens built to change with the people who own them



# Our Mission

"People-first DAO on a mission to help those we serve achieve financial freedom in a secure, smart, and efficient way."

# Introducing



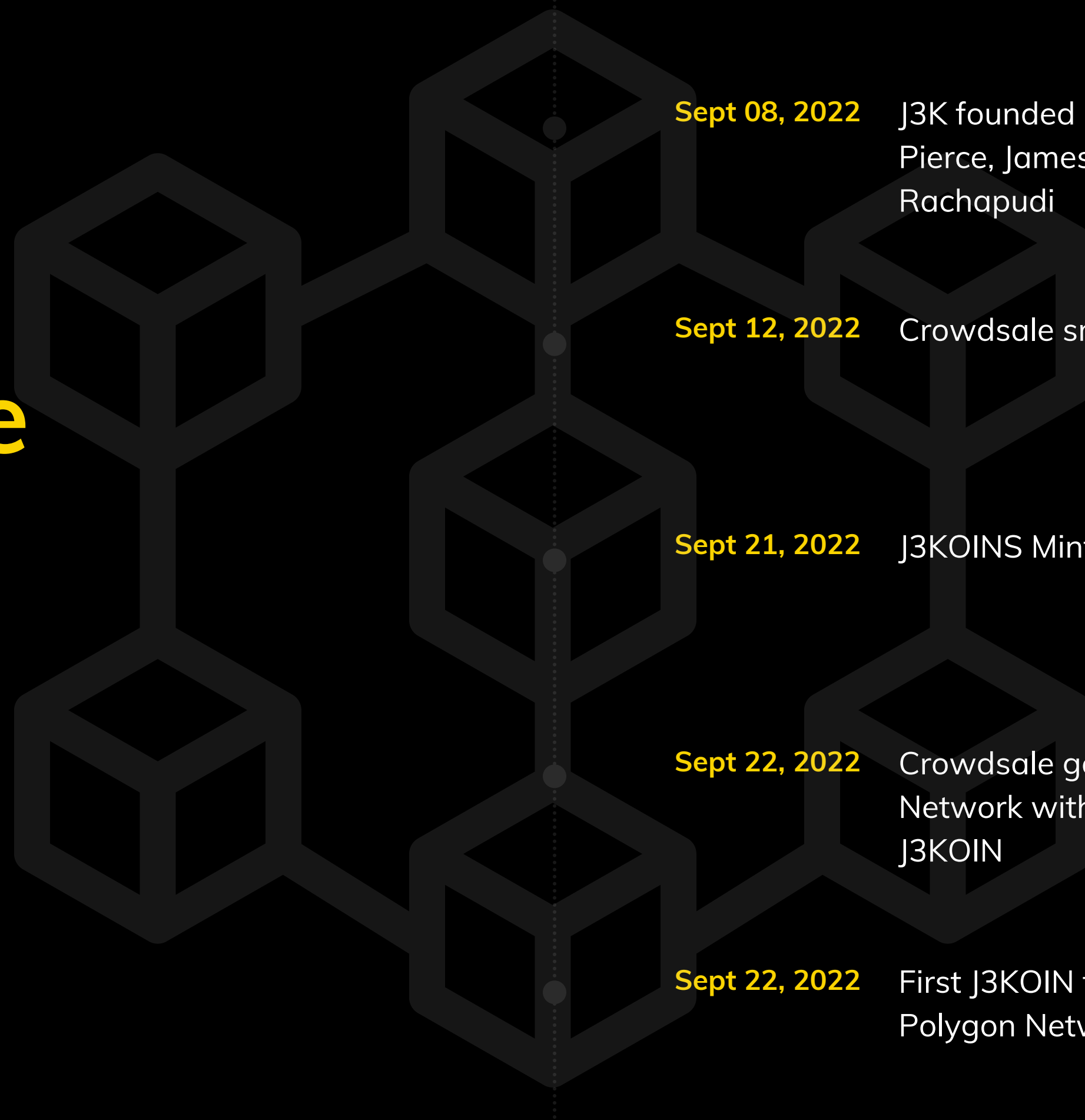
Fungible token designed to help people globally bridge assets and communities

Solidity based Smart contracts utilizing blockchain and the polygon network.

First Edition J3KOIN



# The Crowdsale



**Sept 08, 2022** J3K founded by Jay Wiley, Jeremy Pierce, James Willis, and Kalyan Rachapudi

**Sept 12, 2022** Crowdsale smart contract created

**Sept 21, 2022** J3KOINS Minted by J3K

**Sept 22, 2022** Crowdsale goes live on the Polygon Network with the first version of J3KOIN

**Sept 22, 2022** First J3KOIN transaction on the Polygon Network

# Crowdsale Contract



```
1 pragma solidity ^0.5.0;
2
3 import "../J3Koin_mintable.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 // Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
9 // * Crowdsale
10 // * MintedCrowdsale
11 contract J3KoinCrowdsale 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(
15         uint rate,
16         address payable wallet,
17         J3Koin token
18     ) public Crowdsale(rate, wallet, token) {
19         // constructor can stay empty
20     }
21 }
22
23
24
25 contract J3KoinCrowdsaleDeployer {
26     // Create an `address public` variable called `kasei_token_address`.
27     address public j3koin_token_address;
28     // Create an `address public` variable called `kasei_crowdsale_address`.
29     address public j3koin_crowdsale_address;
30     //create a secondary minter
31
32     // Add the constructor.
33     constructor(
34         string memory name,
35         string memory symbol,
36         address payable wallet
37     ) public {
38         // Create a new instance of the J3Koin contract.
39         J3Koin token = new J3Koin(name, symbol, 0);
40
41         // Assign the token contract's address to the `kasei_token_address` variable.
42         j3koin_token_address = address(token);
43
44         // Create a new instance of the `J3KoinCrowdsale` contract
45         J3KoinCrowdsale j3koin_crowdsale = new J3KoinCrowdsale(1, wallet, token);
46
47         // Assign the `J3KoinCrowdsale` contract's address to the `kasei_crowdsale_address` variable.
48         j3koin_crowdsale_address = address(j3koin_crowdsale);
49
50         //j3koin_wallet_minter_address = address
51         // Set the `J3KoinCrowdsale` contract as a minter
52         token.addMinter(j3koin_crowdsale_address);
53
54         //Adding a wallet of our designation for minting
55
56         // Have the `J3KoinCrowdsaleDeployer` renounce its minter role.
57         token.renounceMinter();
58     }
59 }
```

# Token Contract

```
// * `ERC20`  
// * `ERC20Detailed`  
// * `ERC20Mintable`  
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";  
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";  
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Mintable.sol";  
  
// Create a constructor for the J3Koin contract and have the contract inherit the libraries that you imported from OpenZeppelin.  
contract J3Koin is ERC20, ERC20Detailed, ERC20Mintable {  
    constructor(    infinite gas 1269600 gas  
        string memory name,  
        string memory symbol,  
        uint initial_supply  
    )  
        ERC20Detailed(name, symbol, 18)  
        public  
    {  
        mint(msg.sender, initial_supply);  
    }  
}
```



# Deployed J3K Smart Contract

DEPLOY & RUN  
TRANSACTIONS

ENVIRONMENT  
Injected Provider - Metamask  
Custom (5777) network

ACCOUNT  
0xc37...d8BD2 (99.9986978)

GAS LIMIT  
3000000

VALUE  
0 Wei

CONTRACT  
J3KCrowdsaleDeployer - J3Koin\_crowd

DEPLOY  
NAME: J3Koin  
SYMBOL: J3KN  
WALLET: 0xc37F1f32DDE06c9bE7BAA1ab  
Calldata Parameters **transact**  
☐ Publish to IPFS  
OR  
At Address Load contract from Address

Home J3Koin\_crowdsale.sol J3Koin\_mintable.sol

```
1 pragma solidity ^0.5.0;
2
3 import "../J3Koin_mintable.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 // Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
9 // * Crowdsale
10 // * MintedCrowdsale
11 contract J3KoinCrowdsale 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(
15         uint rate,
16         address payable wallet,
17         J3Koin token
18     ) public Crowdsale(rate, wallet, token) {
19         // constructor can stay empty
20     }
21 }
22
23
24
25 contract J3KoinCrowdsaleDeployer {
26     // Create an `address public` variable called `kasei_token_address`.
27     address public j3koin_token_address;
28     // Create an `address public` variable called `kasei_crowdsale_address`.
29     address public j3koin_crowdsale_address;
30     //create a secondary minter
31
32     // Add the constructor.
33     constructor(
34         string memory name,
35         string memory symbol,
36         address payable wallet
```



# Purchasing J3KOIN

DEPLOY & RUN TRANSACTIONS

ENVIRONMENT  
Injected Provider - Metamask  
Custom (5777) network

ACCOUNT  
0xc37...d8BD2 (99.967798%)

GAS LIMIT  
3000000

VALUE  
15 Ether

CONTRACT  
J3Koin - J3Koin\_mintable.sol

DEPLOY  
NAME: string  
SYMBOL: string  
INITIAL\_SUPPLY: uint256  
Calldata Parameters transact  
Publish to IPFS  
OR  
At Address 0xf8e0d9C3a791E24ec1ad33

Transactions recorded 1

Deployed Contracts  
J3KCROWDSALEDEPLOYER AT 0x0  
Balance: 0 ETH

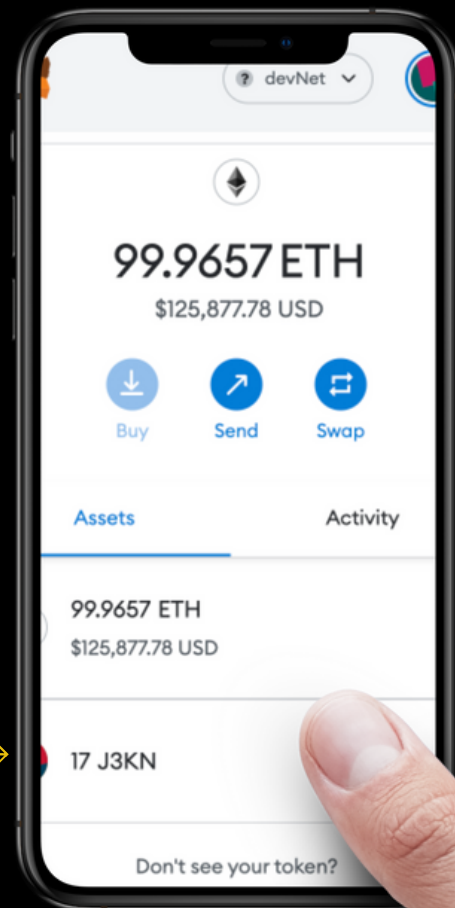
Home J3Koin\_crowdsale.sol J3Koin\_mintable.sol

```
1 pragma solidity ^0.5.0;
2
3 import "../J3Koin_mintable.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 // Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
9 // + Crowdsale
10 // + MintedCrowdsale
11 contract J3KoinCrowdsale 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(
15         uint rate,
16         address payable wallet,
17         J3Koin token
18     ) public Crowdsale(rate, wallet, token) {
19         // constructor can stay empty
20     }
21 }
22
23
24
25 contract J3KCrowdsaleDeployer {
26     // Create an `address public` variable called `kasei_token_address`.
27     address public j3koin_token_address;
28     // Create an `address public` variable called `kasei_crowdsale_address`.
29     address public j3koin_crowdsale_address;
30     //create a secondary minter
31
32     // Add the constructor.
33     constructor(
34         string memory name,
35         string memory symbol,
36         address payable wallet
37     ) public {
38         // Create a new instance of the J3Koin contract.
39         J3Koin token = new J3Koin(name, symbol, 0);
40
41         // Assign the token contract's address to the `kasei_token_address` variable.
42         j3koin_token_address = address(token);
43
44         // Create a new instance of the `J3KoinCrowdsale` contract
45         J3KoinCrowdsale j3koin_crowdsale = new J3KoinCrowdsale(1, wallet, token);
46
47         // Assign the `J3KoinCrowdsale` contract's address to the `kasei_crowdsale_address` variable.
```



# Account Transfer

## Account Balance



The central screen is titled 'Send Tokens' and shows the transfer process. It includes a recipient selection (Account 4), asset selection (J3KN), amount input (10 J3KN), and gas settings (Gas Price: 20 GWEI, Gas Limit: 76692). The screen has 'Cancel' and 'Next' buttons at the bottom.

devNet

Send Tokens

Account 4  
0xb3862a3fab6fe67713e154720145326c9063a75b

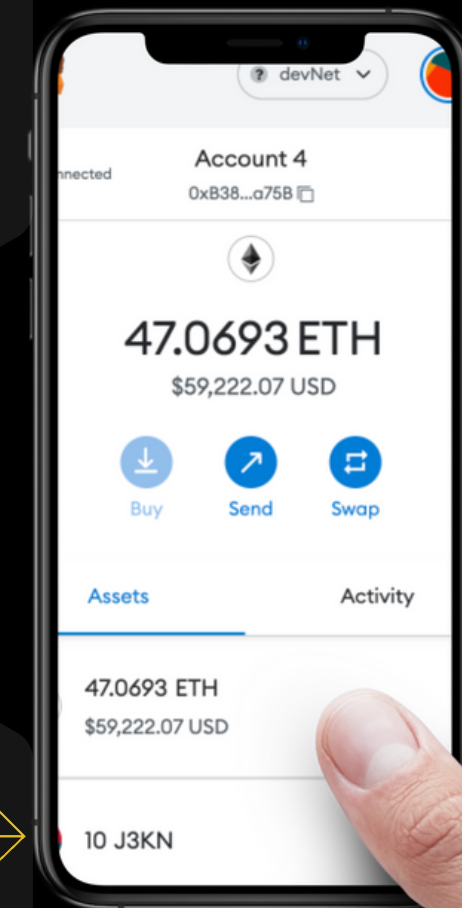
Asset: J3KN  
Balance: 17 J3KN

Amount: 10 J3KN  
No Conversion Rate Available

Gas Price (GWEI) 20 Gas Limit 76692

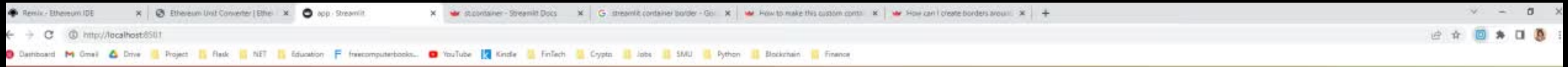
Cancel Next

## Account Recieve





# J3K Client Experience



## Deployer Contract Details

Deployer Contract Address : 0xEf0b6bB171fAF8224F6343Df638D2b75d4e2f586

Crowdsale Contract Address : 0xBd5cd3021EC3c40EF86603De58F12A2aeA5E5B3F

Token Contract Address : 0xEcd0f180e5a41AeA6ce4B6C3d59b48766DBA09cC7

## Crowdsale Contract Details

Crowdsale Contract Address : 0xBd5cd3021EC3c40EF86603De58F12A2aeA5E5B3F

Crowdsale Contract Rate : 1

Crowdsale Contract Token : 0xEcd0f180e5a41AeA6ce4B6C3d59b48766DBA09cC7

Crowdsale Contract Wallet : 0xe7d497a31475902794E70d4c5A04d4f4158baF14



# Polygon Network

Polygon combines the best of Ethereum and other blockchain platforms without the additional limitations, giving J3Koin those same benefits.

	Sidechains	Sharding	Quorum	Cosmos	Polkadot	Polygon
Ethereum Compatibility	● ⚠	● ⚠	●	● ⚠		●
Scalability	●	● ⚠	●	●	● ⚠	●
Security		●			●	●
Sovereignty	●		●	●	● ⚠	●
Interoperability		●		● ⚠	●	●
User Experience	●		●	●	●	●
Developer Experience		● ⚠				●



**Modular Security**

**Interoperability**

**Scalability**

**Low Transaction Fees**

# J3KOIN IS LIVE!!!

DEPLOY & RUN TRANSACTIONS

Transactions recorded 8

Deployed Contracts

J3Koin AT 0xFDA...341D2 (BLOCK)

Balance: 0 ETH

addMinter address account

approve address spender, uint256 amo

decreaseA... address spender, uint256 subtr

increaseAL... address spender, uint256 adde

mint address account, uint256 amou

renounce... address account

transfer address recipient, uint256 amo

transferFr... address sender, address recipie

allowance address owner, address spendi

balanceOf address account

decimals

isMinter address account

```
1 pragma solidity ^0.5.0;
2
3 import "../J3Koin_mintable.sol";
4 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/rel
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/rel
6
7
8 // Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
9 // * Crowdsale
10 // * MintedCrowdsale
11 contract J3KoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE C
12
13 // Provide parameters for all of the features of your crowdsale, such
14
15 creation of J3Koin pending...
```

[block:33437624 txIndex:39] from: 0xA0B...9a1D3 to: value: 0 wei data: 0x608...00000

status true Transaction mined and execution succeed

transaction hash 0xc4a6b1370460a8d2caeba650fb17550085c67acac68a02fcdc21ed16638ec803

from 0xA0B1c445773Ce21440450093359E98E9fc9a1D3

gas 1585454 gas

transaction cost 1585454 gas

input 0x608...00000

decoded output -

val 0 wei

Polygon by Ankr Protocol

0xA0B...a1D3

4.9492 MATIC

Buy Send Swap

Assets Activity

4.9492 MATIC

1000000 J3K

Don't see your token? Import tokens

Polygon Mainnet

4.947

Buy

Assets

Send J3K Sep 22 • To: 0xc2

Contract De Sep 22 • remix.e

0 MATIC \$0.00 USD

0.001 J3K



# The Future

1

Develop a J3K Dex

2

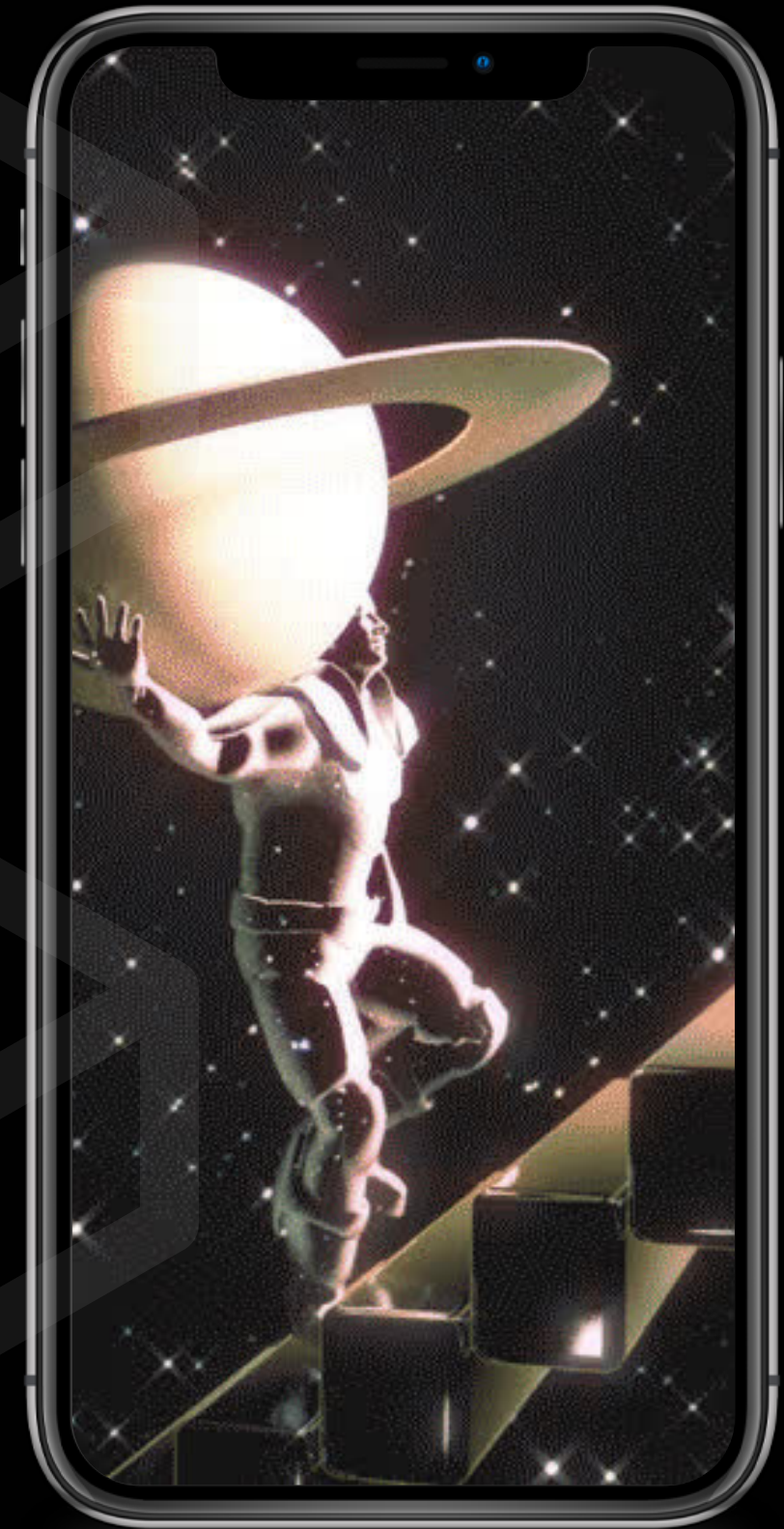
Enhanced Smart Contract design

3

Client Experience Improvements

4

3-5 minute How To educational videos



**Do you  
have any  
questions?**

We are glad to help!

