



Project 3 – Bridge Revolution —————
Faster, and Stable, Low transaction Fees

Collaborators: Whitney Turner, Christy Dain, Racim Bads, Samuel Simon, Dean Kim



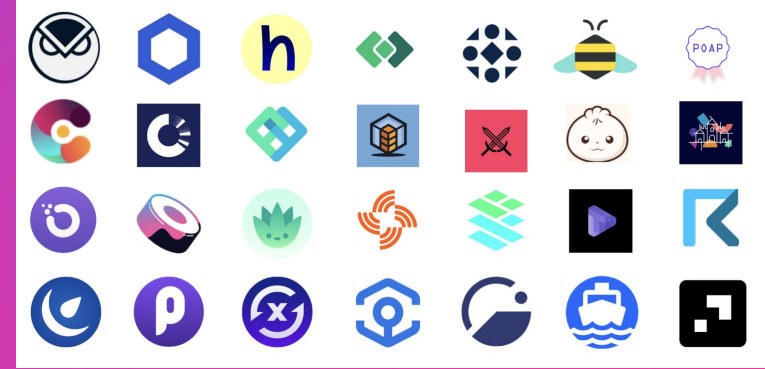
Why xDai ?

- Fast transaction times (5 seconds) & low transaction fees
- Digital cash. A stable chain is ideal for real world value exchange where 1 xDai = 1 US Dollar.
- A stable token for transactions & gas fees.
- A green, energy-efficient and ecologically aware blockchain network.
- Permissionless delegated Proof-Of-Stake based consensus with public POSDAO.
- STAKE token for community consensus participation and incentives.
- Wide-ranging Community Support (see xDai Validator Organizations).
- Extreme usability with tools like Burner Wallet & Burner Wallet 2.
- Growing ecosystem designed to support stable person-to-person transactions, micro transactions, conference currencies, community currencies, DeFi, NFTs, DAOs, games and more.
- Full-featured [BlockScout Explorer](#).
- On-chain, decentralized Random Number Generator.
- Smart Contract, DApp & toolset compatibility with other Ethereum-based chains like Ethereum, Ethereum Classic, BSC and others.

xDai

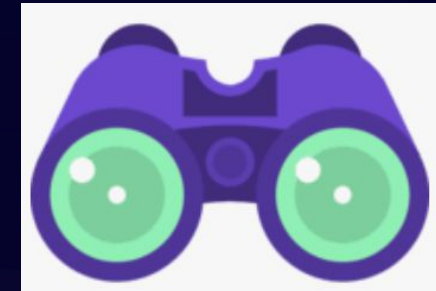
Stable Chain

The xDai chain is a stable payments EVM (Ethereum Virtual Machine) blockchain designed for fast and inexpensive transactions. The chain uses a unique dual-token model; xDai is a stable token used for transactions, payments, and fees, and STAKE is a governance token used to support the underlying POSDAO Proof-of-Stake consensus.



The Tools Used to Compile, Deploy and Interact with The Contract

- Online Vyper Compiler (EtherScan)
- MyEtherWallet (Deploy and Interact)
- BlockScout (easy access to the contract details)
- Vyper Language



Compiling ERC 20 FOTI and MONKE contract

Etherscan

Eth: \$2,908.57 (-6.26%) | 55 Gwei

All Filters

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

HomeBlockchainTokensResourcesMoreSign In

Vyper Online Compiler (Experimental)

Compiles Vyper source code and outputs the ABI, ByteCode and Runtime Bytecode

Step 1 | Select Compiler Version

vyper0.2.12

Step 2 | Enter Source Code Below

```
from vyper.interfaces import ERC20

implements: ERC20

event Transfer:
    sender: indexed(address)
    receiver: indexed(address)
    value: uint256

event Approval:
    owner: indexed(address)
```

Try: Try compiling the sample [Prismaticlabs Contract](#) (for vyper version 0.1.0b7)

Compile VyperReset

Compiler Output

Abi

```
[{"type": "address", {"name": "_value", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 37791}, {"stateMutability": "nonpayable", "type": "function", "name": "mint", "inputs": [{"name": "_to", "type": "address"}, {"name": "_value", "type": "uint256"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 79541}, {"stateMutability": "nonpayable", "type": "function", "name": "burn", "inputs": [{"name": "_value", "type": "uint256"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 77898}, {"stateMutability": "nonpayable", "type": "function", "name": "burnFrom", "inputs": [{"name": "_to", "type": "address"}, {"name": "_value", "type": "uint256"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 115549}, {"stateMutability": "view", "type": "function", "name": "name", "inputs": [{"name": ""}], "outputs": [{"name": ""}], "gas": 12878}, {"stateMutability": "view", "type": "function", "name": "symbol", "inputs": [{"name": ""}], "outputs": [{"name": ""}], "gas": 10623}, {"stateMutability": "view", "type": "function", "name": "decimals", "inputs": [{"name": ""}], "outputs": [{"name": ""}], "gas": 2628}, {"stateMutability": "view", "type": "function", "name": "balanceOf", "inputs": [{"name": "arg0", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 2873}, {"stateMutability": "view", "type": "function", "name": "allowance", "inputs": [{"name": "arg0", "type": "address"}, {"name": "arg1", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 3118}, {"stateMutability": "view", "type": "function", "name": "totalSupply", "inputs": [{"name": ""}], "outputs": [{"name": "", "type": "uint256"}], "gas": 2718}]
```

Bytecode

This website uses cookies to improve your experience and has an updated Privacy Policy.

Got it

FOTI Contract

Deploying the ERC 20 FOTI and Monke contract

- Copy the Abi and Bytecode from the compiled contract and paste them in the V5.MyEtherWallet deployer and sign transaction

[illegible]

Swap

DApps

Contract

Interact with Contract

Deploy Contract

Message

Deploy Contract

Byte Code

Clear Copy

ABI/JSON Interface

Clear Copy

Contract Name

Name for the contract

Sign Transaction

Clear All

A screenshot of a web application interface for interacting with a contract. On the left, a sidebar contains a menu with 'DApps' at the top, followed by 'Contract' (selected), 'Message', and 'ABI/JSON Interface'. The main content area is titled 'Byte Code' and contains a large, empty text input field. Below this is the 'ABI/JSON Interface' section, also with an empty text input field. Further down is the 'Contract Name' section with a text input field containing the placeholder 'Name for the contract'. At the bottom of the main area is a 'Sign Transaction' button. A modal dialog is open in the center, featuring a green checkmark icon, the word 'Success', and the text 'Continue transaction with Web3 Wallet Provider.' with a 'Close' button. The background is a light gray.

FOTI and Monke contract Addresses

Address Details

contract

token



0x4033644EAF6ca53C1b4666EF0d924dEB4d4C6

Vyper_contract

FOTI
(FOTI)

9 Transactions

309,999 Gas used

Last Balance Update:

Block #18,262,819

Created by 0x55692c-718ded at 0xf8fb3d-b3be78

Address Details

contract

token



0x44aD7F020A488629BF79D20cBE5833b1A7C8243E

Vyper_contract

Monke
(MONKE)

11 Transactions

647,324 Gas used

Last Balance

Update: Block #18,262,819

Created by 0x55692c-718ded at 0xd3caff-fd633f

BlockScouts

- Once deployed, The contracts can be accessed from metamask through blockscouts

Address Details contract token

0xaEE4033644EAF6ca53C1b4666FF0d924dEB4d4C6

Vyper_contract

FOTI

(FOTI)

9 Transactions
309,999 Gas used
Last Balance Update:
Block #18,256,709

Created by 0x55692c-718ded at 0xf8fb3d-b3be78

Balance

0 xDAI

\$0.000000 USD (@ \$1.00/xDAI)

0 tokens

Transactions

Internal Transactions

Coin Balance History

Logs

Code ✓

Read Contract

Write Contract

• Connected to 0x55692cc9781c300e0a56ba3a67ba057c8c718ded

1. transfer → + Write

↳ bool

2. transferFrom → + Write

↳ bool

3. approve → + Write

↳ bool

4. mint → + Write

5. burn → + Write

FOTI Contract

Address Details contract token

0x44aD7E020A488629BF79D20cBE5833b1A7C8243E

Vyper_contract

Monke

(MONKE)

9 Transactions
537,340 Gas used
Last Balance
Update: Block #18,256,789

Created by 0x55692c-718ded at 0xd3caff-fd633f

Balance

0 xDAI

\$0.000000 USD (@ \$1.00/xDAI)

1 token

Transactions

Token Transfers

Tokens

Internal Transactions

Coin Balance History

Logs

Code ✓

Read Contract

Write Contract

• Connected to 0x55692cc9781c300e0a56ba3a67ba057c8c718ded

1. transfer → + Write

↳ bool

2. transferFrom → + Write

↳ bool

3. approve → + Write

↳ bool

4. monk → Write

bool

5. monk → + Write

↳ bool

Monke Contract

Interacting with the contracts through MyEtherWallet (Approve)

You are first asked to input the Contract address and the ABI

Interact with Contract

Contract Address

Select an item



ABI/JSON Interface

Clear

Copy

Continue



Clear All

Read / Write Contract

Contract Address: 0xaEE4033644EAF6ca53C1b4666EF0d924dEB4d4C6

approve



Approve

```
_spender (address):
```

0x44aD7F020A488629BF79D20cBE5833b1A7C8243F



```
_value (uint256):
```

1000000000000000000



Value in ETH:

0

[Back](#)

Write

Clear All

Interacting with the contracts through MyEtherWallet (MONK)

You are first asked to input the Contract address and the ABI

Using MyEtherWallet we can now monk our FOTI

Swap

DApps

Contract

Interact with Contract

Deploy Contract

Message

Interact with Contract

Contract Address Select an item

Enter Contract Address

ABI/JSON Interface Clear Copy

Continue →

Clear All

Swap

DApps

Contract

Interact with Contract

Deploy Contract

Message

Interact with Contract

Read / Write Contract

Contract Address: 0x44aD7F020A488629BF79D20cBE5833b1A7C8243F monk

Monk

Amount (uint256):

6000000000000000000

Value in ETH:

0

Back Write

Clear All

MONK

Interacting with the contracts through MyEtherWallet

Transaction Details	
Open in Tenderly	
Transaction Hash	0xbcafe5d568eef42f042833ae526c109a4dad7b961aa16f514d8fb2d72c8de24d
Result	Success
Status	Confirmed Confirmed by 2 block
Block	18257213
Timestamp	13 seconds ago September-24-2021 11:30:45 PM +-4 UTC Confirmed within 603-976 milliseconds
From	0x55692cc9781c300e0a56ba3a67ba057c8c718ded
Interacted With (To)	0x44ad7f020a488629bf79d20cbe5833b1a7c8243f
Tokens Transferred	From 0x55692c-718ded To Vyper_contract (0x44ad7f-c8243f) For 6 FOTI
Tokens Minted	From 0x000000-000000 To 0x55692c-718ded For 6,000,000 MONKE
Value	0 xDAI (\$0.000000 USD)
Transaction Fee	0.00005527299 xDAI (\$0.000055 USD)
Gas Price	1.005 Gwei
Gas Limit	59,722
Gas Used by Transaction	54,998 92.09%
	90 22

Transact 0.6FOTI for 600,000 MONKE

Dapp.js

The Dapp.js is function here is to create a User interface where the Foti tokens can be exchanged for large amount of Monke

We do this by importing the abis from the foti, monke and Token farm contracts.

```
1 import React, { Component } from 'react'
2 import Monke from '../src/abis/monke.json'
3 import Navbar from './Navbar'
4 import Main from './Main'
5 import '../src/styles/Global.css'
6 import 'tailwindcss/tailwind.css'
7 import { Web3ReactProvider } from '@web3-react/core'
8 import Web3 from 'web3'
9 function getLibrary(provider) {
10   return new Web3(provider)
11 }
12 function tailwind ({ Component, pageProps }) {
13   return (
14     <Web3ReactProvider getLibrary={getLibrary}>
15       <Component {...pageProps} />
16     </Web3ReactProvider>
17   )
18 }
19 export default MyApp
20 class MyApp extends Component {
21   async componentWillMount() {
22     await this.loadWeb3()
23     await this.loadBlockchainData()
24   }
25 }
26 async ; loadBlockchainData(); {
27   const web3 = window.web3
28   const accounts = await web3.eth.getAccounts()
29   this.setState({ account: accounts[0] })
30   const networkId = await web3.eth.net.getId()
31   // Load MonkeToken
32   const MonkeData = Monke.networks[networkId]
33   if(MonkeData) {
34     const mMonke = new web3.eth.Contract(Monke.abi, MonkeData.address)
35     this.setState({ Monke })
36     let MonkeBalance = await Monke.methods.balanceOf(this.state.account).call()
37     this.setState({ MonkeBalance: MonkeBalance.toString() })
38   } else {
39     window.alert('Monke contract not deployed to detected network.')
40   }
41 }
```

```

43 async ; loadWeb3() ; {
44   if (window.ethereum) {
45     window.web3 = new Web3(window.ethereum)
46     await window.ethereum.enable()
47   }
48   else if (window.web3) {
49     window.web3 = new Web3(window.web3.currentProvider)
50   }
51   else {
52     window.alert('Non-Ethereum browser detected. You should consider trying MetaMask!')
53   }
54 }
55 stakeTokens = (amount) => {
56   this.setState({ loading: true })
57   this.state.Foti.methods.approve(this.state.monke._address, amount).send({ from: this.state.account }).on('transactionHash', (hash) => {
58     this.state.monke.methods.stakeTokens(amount).send({ from: this.state.account }).on('transactionHash', (hash) => {
59       this.setState({ loading: false })
60     })
61   })
62 }
63 unstakeTokens = (amount) => {
64   this.setState({ loading: true })
65   this.state.monke.methods.unstakeTokens().send({ from: this.state.account }).on('transactionHash', (hash) => {
66     this.setState({ loading: false })
67   })
68 }
69 constructor(props) ; {
70   super(props)
71   this.state = {
72     account: '0x0',
73     Monke: {},
74     Foti: {},
75     FotiBalance: '0',
76     monkeBalance: '0',
77     stakingBalance: '0',
78     loading: true
79   }
80 }

```

Dapp.js

Here in the Dapp we are creating the circumstances in which Foti can be staked by Monke for exchange/conversion

- Dapp.js continue....
-
-
-

```

81   render() ; {
82     let content
83     if(this.state.loading) {
84       content = <p id="loader" className="text-center">Loading...</p>
85     } else {
86       content = <Main
87         monkeBalance={this.state.monkeBalance}
88       />
89     }
90     return (
91       <div>
92         <Navbar account={this.state.account} />
93         <div className="container-fluid mt-5">
94           <div className="row">
95             <main role="main" className="col-lg-12 ml-auto mr-auto" style={{ maxWidth: '600px' }}>
96               <div className="content mr-auto ml-auto">
97                 <a
98                   href=""
99                   target="_blank"
100                   rel="noopener noreferrer"
101                 >
102                   </a>
103                 {content}
104               </div>
105             </main>
106           </div>
107         </div>
108       </div>
109     );
110   }

```

Dapp.js

We are importing the final touches on the interface here so that the website is compiled and looking like a normal website

Index.js

We are using Web3 injected connector here to connect your MetaMask wallet to the UI via react button

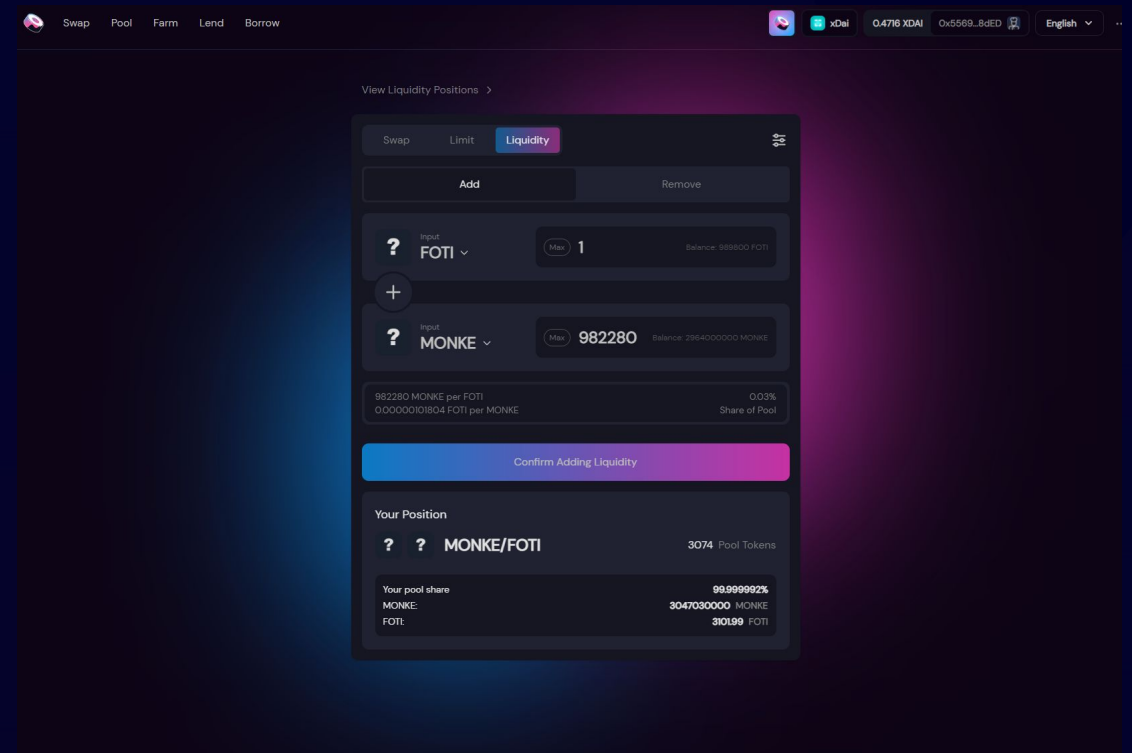
```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import { useWeb3React } from "@web3-react/core";
4 import Dapp from '../src/Dapp.js';
5 import * as serviceWorker from './serviceWorker';
6 import { injected } from "../src/connector.js";
7 export default function Home() {
8   const { active, account, library, connector, activate, deactivate } = useWeb3React()
9   async function connect() {
10     try {
11       await activate(injected)
12     } catch (ex) {
13       console.log(ex)
14     }
15   }
16   async function disconnect() {
17     try {
18       deactivate()
19     } catch (ex) {
20       console.log(ex)
21     }
22   }
23   return (
24     <div className="flex flex-col items
25       <button onClick={connect} classNa
26       {active ? <span>Connected with <b
27       <button onClick={disconnect} clas
28     </div>
29   )
30 }
31 ReactDOM.render(<Dapp />, document.getElementById('root'));
32 // If you want your app to work offline and load faster, you can change
33 // unregister() to register() below. Note this comes with some pitfalls.
34 // Learn more about service workers: https://bit.ly/CRA-PWA
35 serviceWorker.unregister();
```

```
1 import { InjectedConnector } from '@web3-react/injected-connector'
2 export const injected = new InjectedConnector({
3   supportedChainIds: [1, 3, 4, 5, 42, 100],
4 })
```

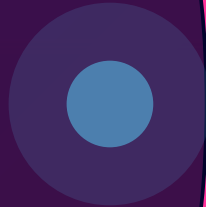
Creating the interface for the exchange

```
interface UniswapExchangeInterface():
    # Public Variables
    def tokenAddress() -> address: 0xaEE4033644EAF6ca53C1b4666EF0d924dEB4d4C6
    def factoryAddress() -> address: 0x44aD7F020A488629BF79D20cBE5833b1A7C8243F
    # Providing Liquidity
    def addLiquidity(min_liquidity: uint256, max_tokens: uint256, deadline: timestamp) -> uint256: modifying
    def removeLiquidity(amount: uint256, min_eth: uint256(wei), min_tokens: uint256, deadline: timestamp) -> uint256: modifying
    # Trading
    def ethToTokenSwapInput(min_tokens: uint256, deadline: timestamp) -> uint256: modifying
    def ethToTokenTransferInput(min_tokens: uint256, deadline: timestamp, recipient: address) -> uint256: modifying
    def ethToTokenSwapOutput(tokens_bought: uint256, deadline: timestamp) -> uint256(wei): modifying
    def ethToTokenTransferOutput(tokens_bought: uint256, deadline: timestamp, recipient: address) -> uint256: modifying
    def tokenToEthSwapInput(tokens_sold: uint256, min_eth: uint256(wei), deadline: timestamp) -> uint256: modifying
    def tokenToEthTransferInput(tokens_sold: uint256, min_eth: uint256(wei), deadline: timestamp) -> uint256: modifying
    def tokenToEthSwapOutput(eth_bought: uint256(wei), max_tokens: uint256, deadline: timestamp) -> uint256: modifying
    def tokenToEthTransferOutput(eth_bought: uint256(wei), max_tokens: uint256, deadline: timestamp) -> uint256: modifying
    def tokenToTokenSwapInput(tokens_sold: uint256, min_tokens_bought: uint256, min_eth_bought: uint256(wei)) -> uint256: modifying
    def tokenToTokenTransferInput(tokens_sold: uint256, min_tokens_bought: uint256, min_eth_bought: uint256(wei)) -> uint256: modifying
    def tokenToTokenSwapOutput(tokens_bought: uint256, max_tokens_sold: uint256, max_eth_sold: uint256(wei)) -> uint256: modifying
    def tokenToTokenTransferOutput(tokens_bought: uint256, max_tokens_sold: uint256, max_eth_sold: uint256(wei)) -> uint256: modifying
    def tokenToExchangeSwapInput(tokens_sold: uint256, min_tokens_bought: uint256, min_eth_bought: uint256(wei)) -> uint256: modifying
    def tokenToExchangeTransferInput(tokens_sold: uint256, min_tokens_bought: uint256, min_eth_bought: uint256(wei)) -> uint256: modifying
    def tokenToExchangeSwapOutput(tokens_bought: uint256, max_tokens_sold: uint256, max_eth_sold: uint256(wei)) -> uint256: modifying
    def tokenToExchangeTransferOutput(tokens_bought: uint256, max_tokens_sold: uint256, max_eth_sold: uint256(wei)) -> uint256: modifying
    # Get Price
    def getEthToTokenInputPrice(eth_sold: uint256(wei)) -> uint256: constant
    def getEthToTokenOutputPrice(tokens_bought: uint256) -> uint256(wei): constant
    def getTokenToEthInputPrice(tokens_sold: uint256) -> uint256(wei): constant
    def getTokenToEthOutputPrice(eth_bought: uint256(wei)) -> uint256: constant
    # Pool Token ERC20 Compatibility
    def balanceOf() -> address: constant
    def allowance(_owner : address, _spender : address) -> uint256: constant
    def transfer(_to : address, _value : uint256) -> bool: modifying
    def transferFrom(_from : address, _to : address, _value : uint256) -> bool: modifying
    def approve(_spender : address, _value : uint256) -> bool: modifying
    # Setup
    def setup(token_addr: address): modifying
```

We need to import the abi from this token exchange contract in order to have the swap function in our interface



<https://app.sushi.com/swap>



Thank You

And See You Next Time
