Lesson 16 -

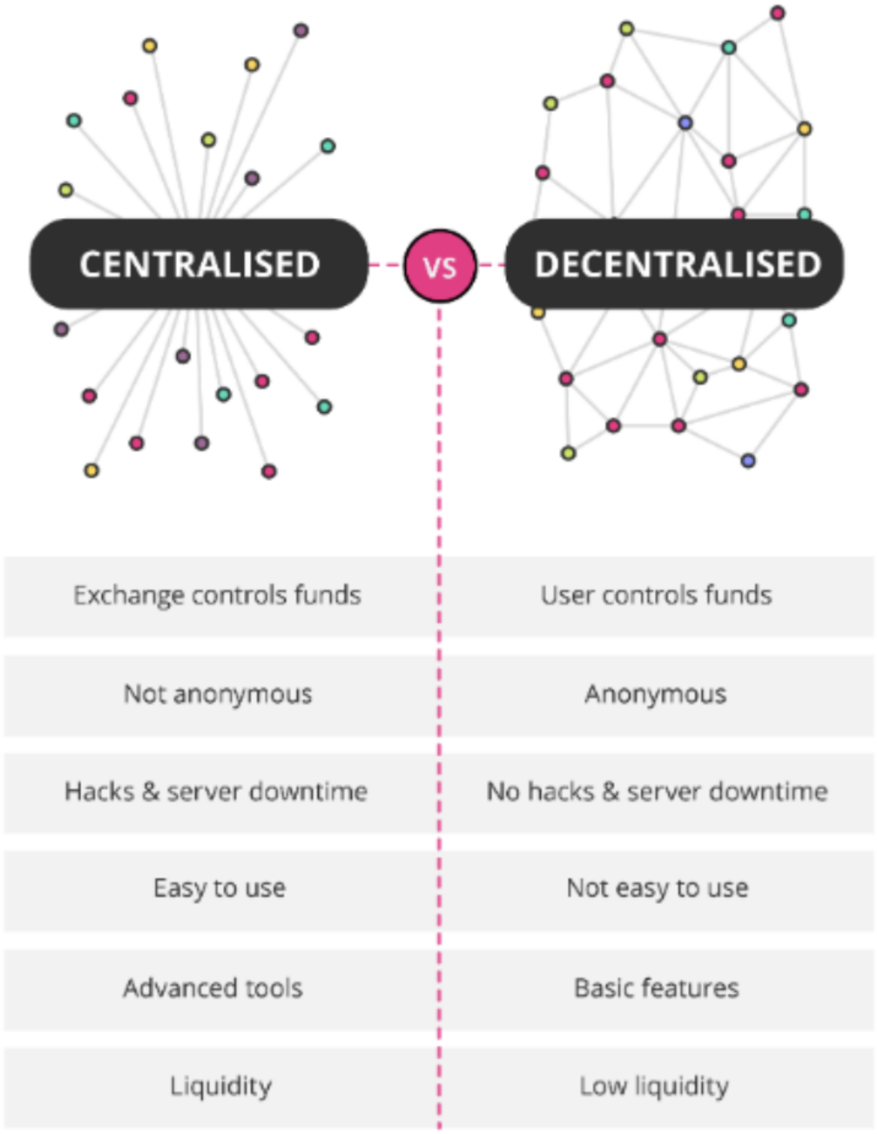
This week

Monday - DeFi continued
Tuesday - Upgradability
Wednesday - Zero Knowledge Proofs
Thursday - Introduction to Security

AMM Review

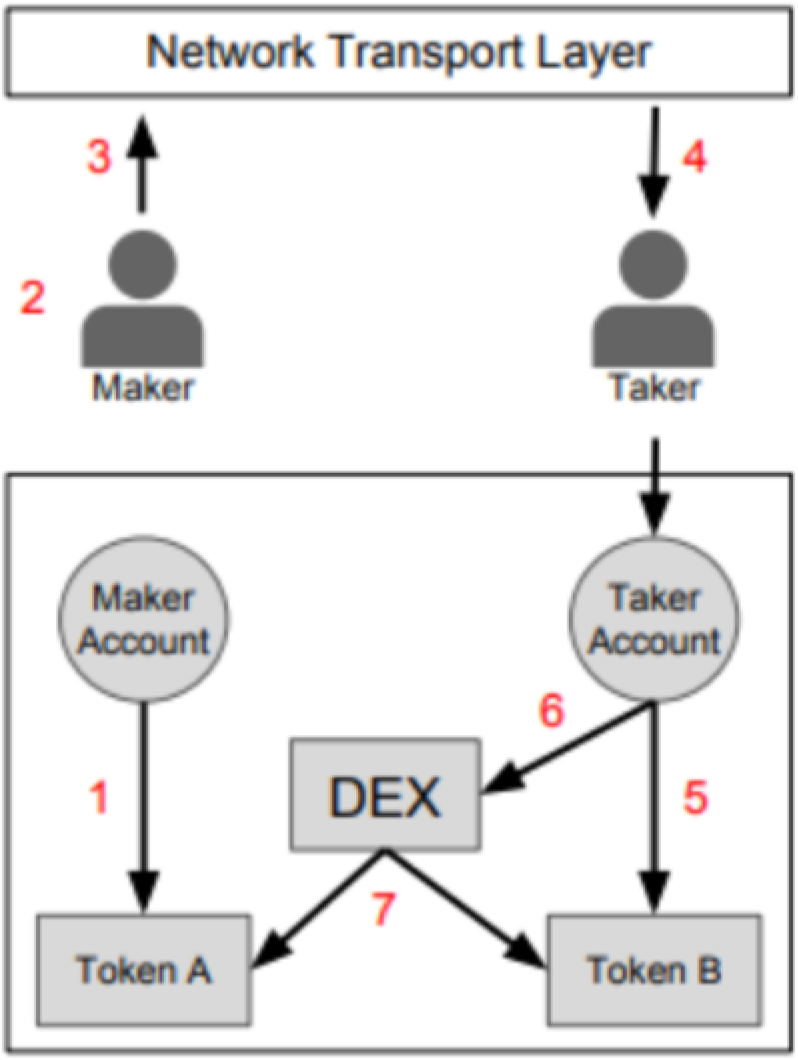
Decentralised Exchanges

Decentralised Exchanges are a protocol to provide asset exchange without the platform holding the users assets
Vitalik “centralised exchanges go burn in hell as much as possible”



CENTRALISED	DECENTRALISED
Exchange controls funds	User controls funds
Not anonymous	Anonymous
Hacks & server downtime	No hacks & server downtime
Easy to use	Not easy to use
Advanced tools	Basic features
Liquidity	Low liquidity

Early Exchanges - 0x Protocol

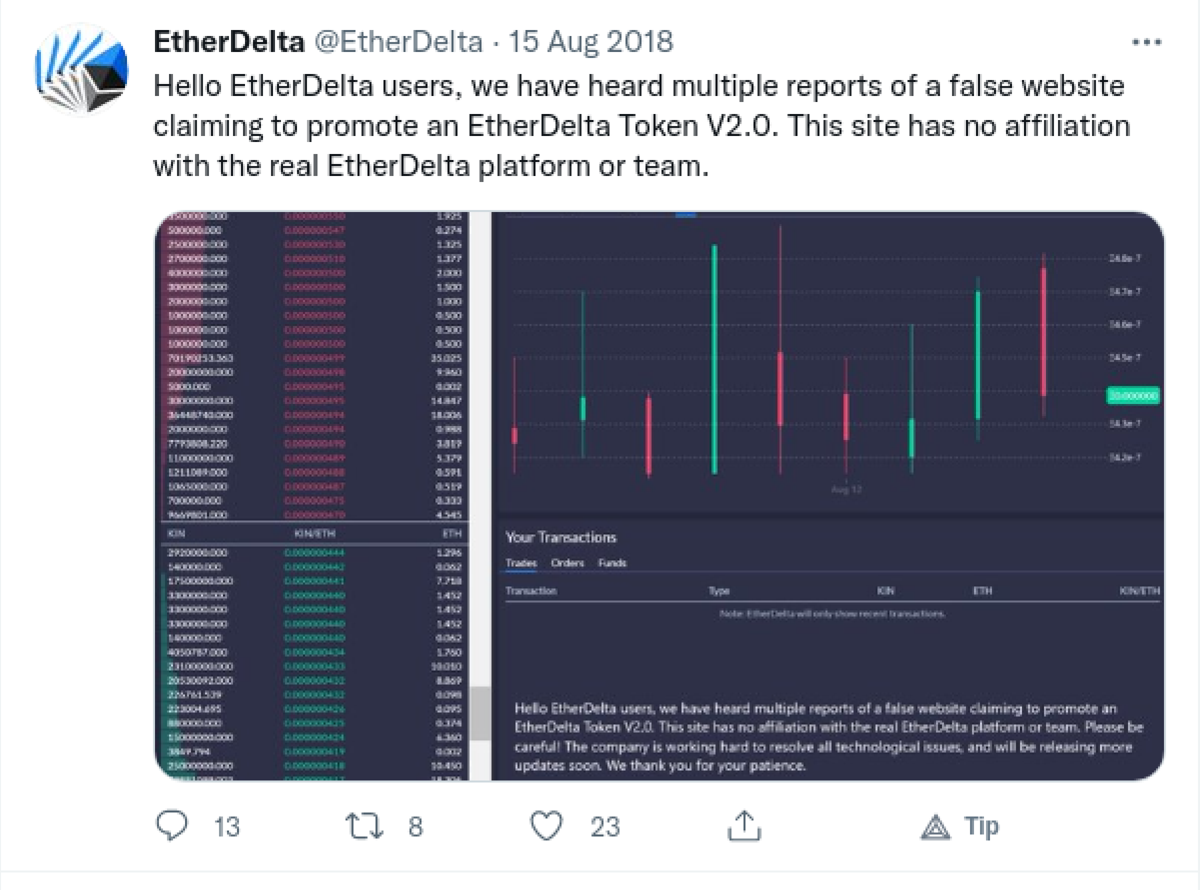


1. Maker approves the decentralized exchange (DEX) contract to access their balance of Token A.
2. Maker creates an order to exchange Token A for Token B, specifying a desired exchange rate, expiration time (beyond which the order cannot be filled), and signs the order with their private key.
3. Maker broadcasts the order over any arbitrary communication medium.
4. Taker intercepts the order and decides that they would like to fill it.
5. Taker approves the DEX contract to access their balance of Token B.
6. Taker submits the makers signed order to the DEX contract. 7. The DEX contract authenticates makers signature, verifies that the order has not expired, verifies that the order has not already been filled, then transfers tokens between the two parties at the specified exchange rate.

EtherDelta

December 2017 Ether Delta is attacked

The DNS for Ether Delta is redirected to a fake site
Many people send tokens to this site thinking it is genuine
308 ETH stolen



EtherDelta @EtherDelta · 15 Aug 2018

Hello EtherDelta users, we have heard multiple reports of a false website claiming to promote an EtherDelta Token V2.0. This site has no affiliation with the real EtherDelta platform or team.

3500000.000	0.000000550	1.925
500000.000	0.000000547	0.274
2500000.000	0.000000530	1.325
2700000.000	0.000000510	1.377
4000000.000	0.000000500	2.000
3000000.000	0.000000500	1.500
2000000.000	0.000000500	1.000
1000000.000	0.000000500	0.500
1000000.000	0.000000500	0.500
1000000.000	0.000000500	0.500
70190253.363	0.000000499	35.025
20000000.000	0.000000498	9.960
5000.000	0.000000495	0.002
30000000.000	0.000000495	14.847
36448740.000	0.000000494	18.006
2000000.000	0.000000494	0.988
7793808.220	0.000000490	3.819
11000000.000	0.000000489	5.379
1211089.000	0.000000488	0.591
1065000.000	0.000000487	0.519
700000.000	0.000000475	0.333
9669801.000	0.000000470	4.545
KIN	KIN/ETH	ETH
2920000.000	0.000000444	1.296
140000.000	0.000000442	0.062
17500000.000	0.000000441	7.718
3300000.000	0.000000440	1.452
3300000.000	0.000000440	1.452
3300000.000	0.000000440	1.452
140000.000	0.000000440	0.062
4050787.000	0.000000434	1.760
23100000.000	0.000000433	10.010
20530092.000	0.000000432	8.869
226761.539	0.000000432	0.098
223004.695	0.000000426	0.095
880000.000	0.000000425	0.374
15000000.000	0.000000424	6.360
3849.794	0.000000419	0.002
25000000.000	0.000000418	10.450

Your Transactions

Transaction	Type	KIN	ETH	KIN/ETH

Note: EtherDelta will only show recent transactions.

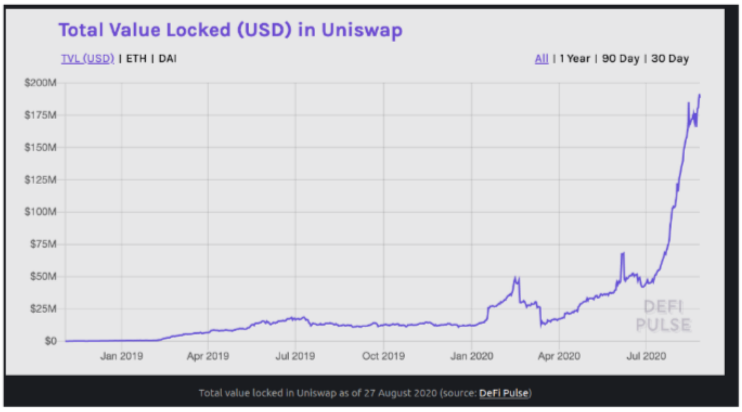
Hello EtherDelta users, we have heard multiple reports of a false website claiming to promote an EtherDelta Token V2.0. This site has no affiliation with the real EtherDelta platform or team. Please be careful! The company is working hard to resolve all technological issues, and will be releasing more updates soon. We thank you for your patience.

Uniswap

The first ideas came from Vitalik, Nick Johnson and Martin Koppelmann in 2016 in a Reddit post

It was followed by an implementation from Hayden Adams and launched in Nov 2018

- Launched in 2018, Uniswap is a DEX featuring an AMM
- Solves the problem of illiquid assets since anyone can set up a liquidity pool



Total value locked in Uniswap as of 27 August 2020 (source: DeFi Pulse)

- Truly Decentralised
- Allows swap between any ERC20 pairs
- The code is robust

V2 Launched May 2020 allowing direct token swaps - halving gas fees

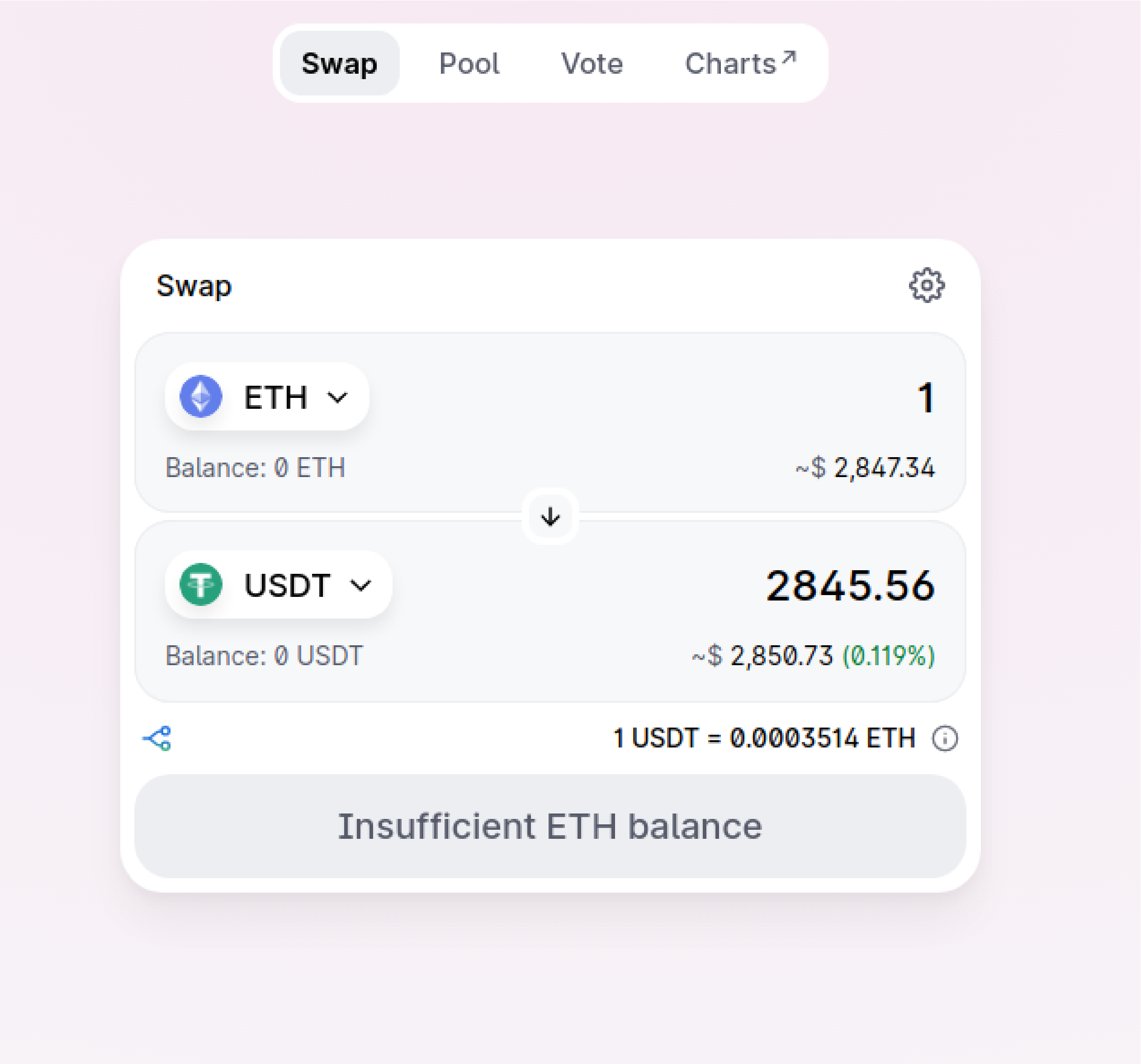
It solved many of the problems of the initial exchanges such as lack of incentives to provide liquidity for rarely traded assets.
It relies on a smart contract acting as an automatic market maker (AMM)

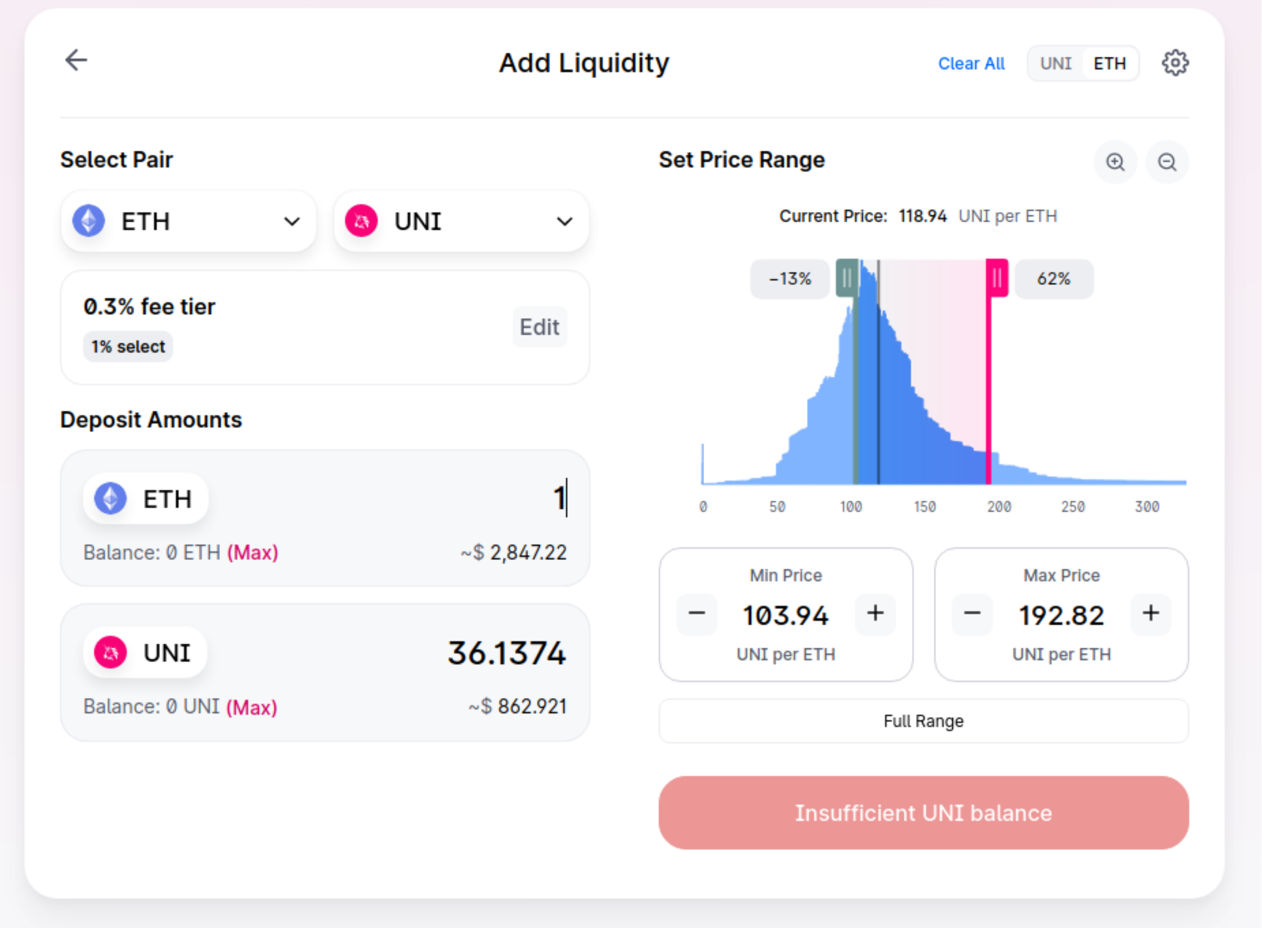
Automatic Market Makers

Incentivising Users

- Users deposit funds into a liquidity pool, for example ETH and USDT
- This pool (a token pair) allows users to exchange (or maybe lend or borrow) tokens
- Interacting with the exchange incurs fees
- These fees are paid to the liquidity providers

They are characterised as constant function market makers.

> From Constant Function Market Makers
>
> The term “constant function” refers to the fact that any trade must change the reserves in such a way that the product of those reserves remains unchanged (i.e. equal to a constant).
>
> 
>
>
>
> ETH 1
>
> Balance: 0 ETH ~$ 2,847.34
>
> USDT 2845.56
>
> Balance: 0 USDT ~$ 2,850.73 (0.119%)
>
> 1 USDT = 0.0003514 ETH



Add Liquidity

Select Pair

ETH UNI

0.3% fee tier

1% select

Deposit Amounts

ETH 1

Balance: 0 ETH (Max) ~$ 2,847.22

UNI 36.1374

Balance: 0 UNI (Max) ~$ 862.921

Set Price Range

Current Price: 118.94 UNI per ETH

Min Price 103.94 UNI per ETH

Max Price 192.82 UNI per ETH

LP Tokens

Typically the liquidity provider receives LP tokens when they add liquidity, say ETH and USDT

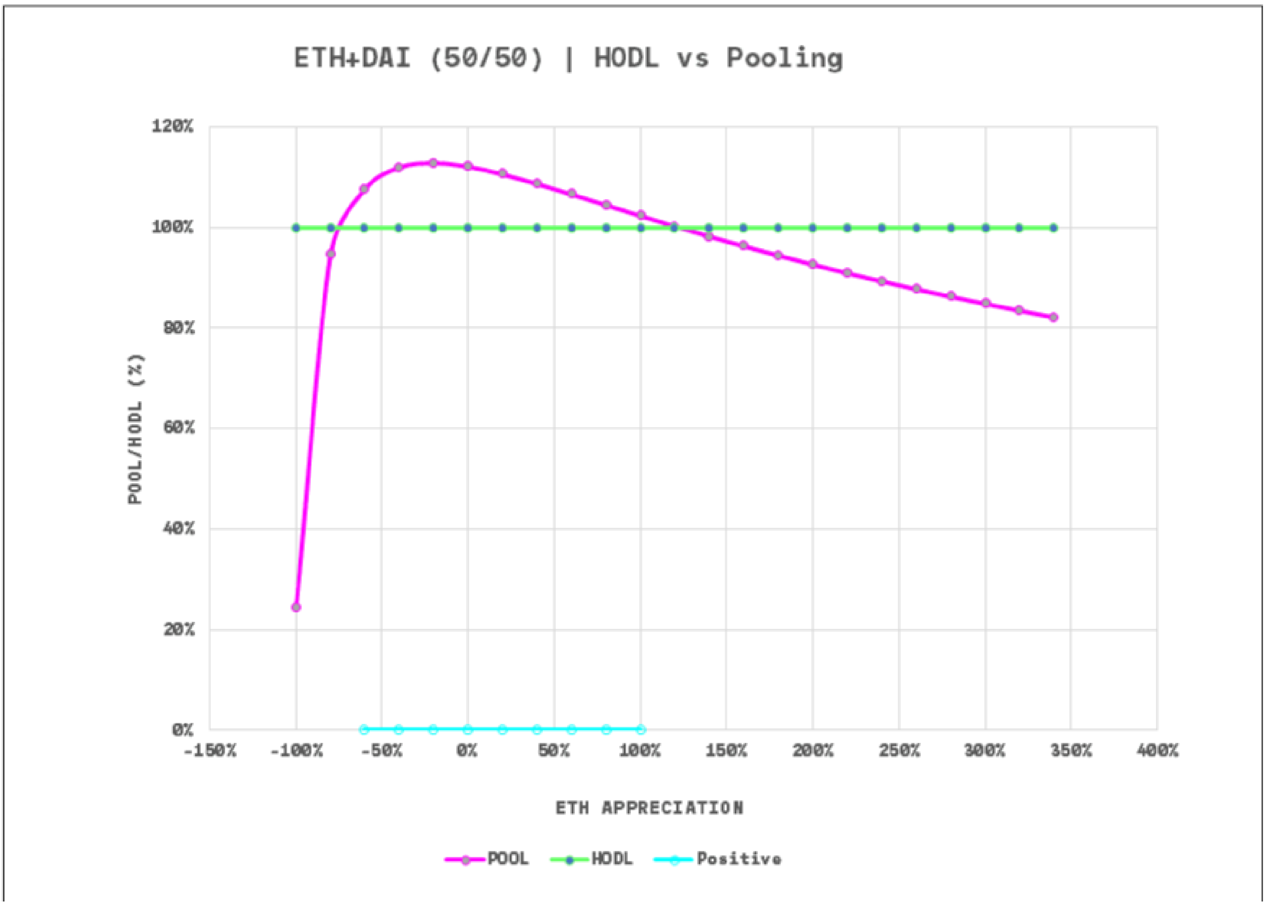
Later they can take liquidity by providing LP tokens to the contract and will receive back ETH and USDT.

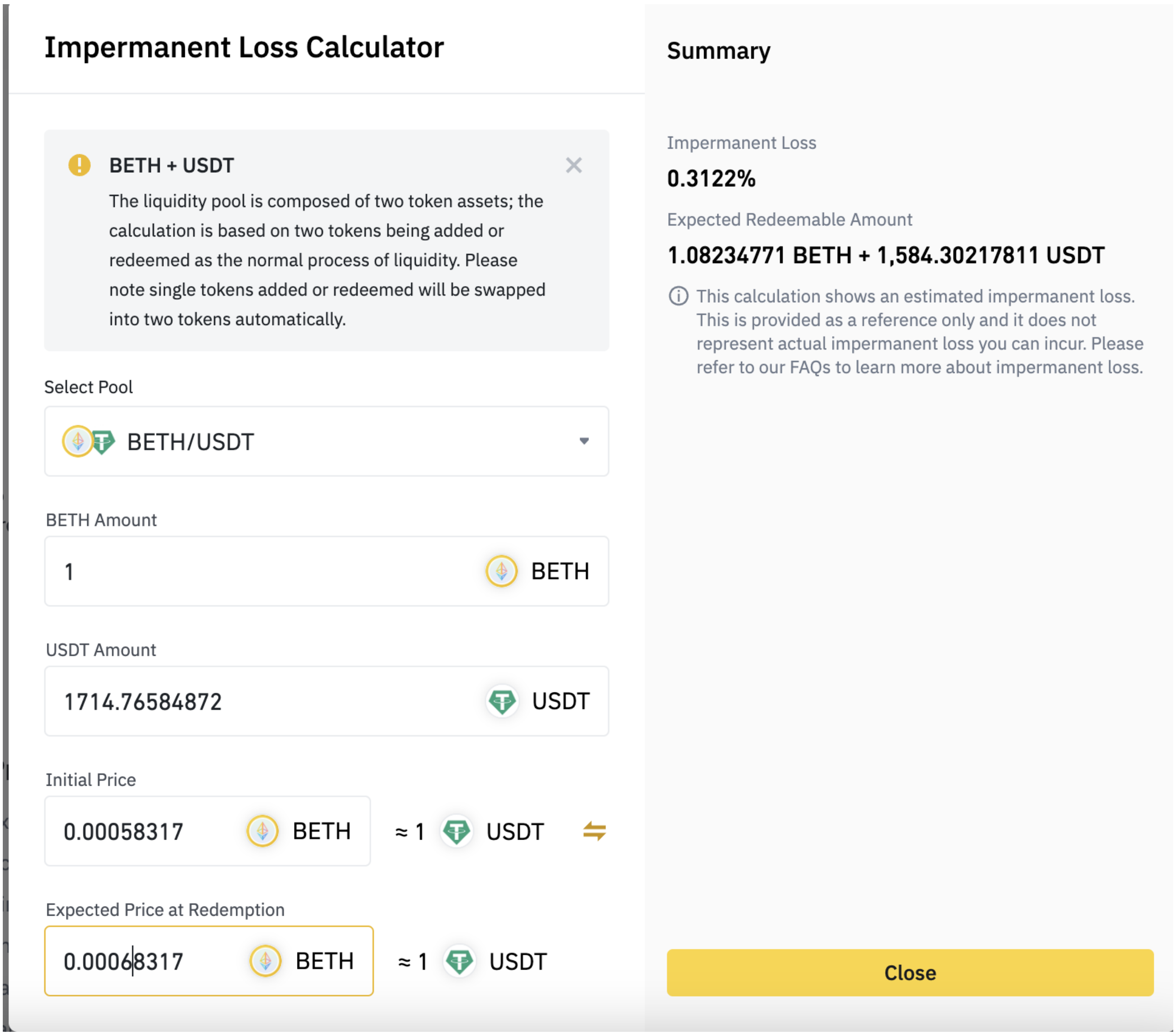
Ideally they will make a profit

Risks associated with AMMs

- Slippage
- Large trades can move the price
- Impermanent loss

**


ETH+DAI (50/50) | HODL vs Pooling
120%
100%
80%
60%
40%
20%
0%
POOL/HODL (%)
-150%
-100%
-50%
0%
50%
100%
150%
200%
250%
300%
350%
400%
ETH APPRECIATION
POOL
HODL
Positive



Impermanent Loss Calculator

BETH + USDT

The liquidity pool is composed of two token assets; the calculation is based on two tokens being added or redeemed as the normal process of liquidity. Please note single tokens added or redeemed will be swapped into two tokens automatically.

Select Pool: BETH/USDT

BETH Amount: 1 BETH

USDT Amount: 1714.76584872 USDT

Initial Price: 0.00058317 BETH ≈ 1 USDT

Expected Price at Redemption: 0.00068317 BETH ≈ 1 USDT

Summary

Impermanent Loss

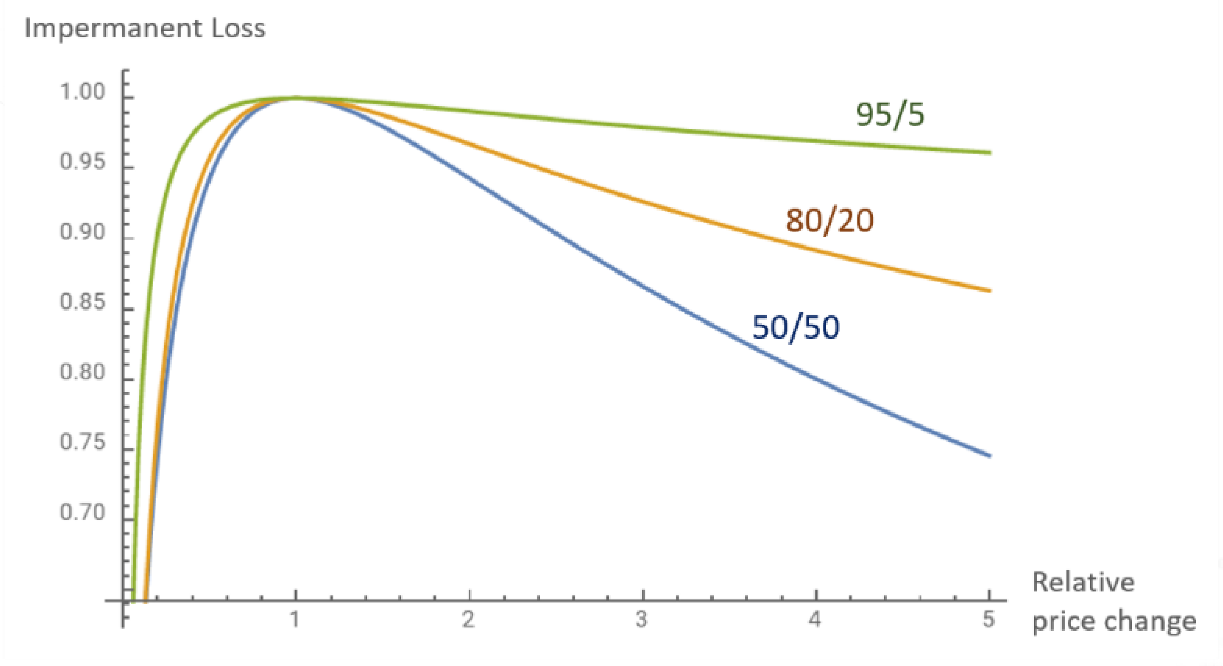
0.3122%

Expected Redeemable Amount

1.08234771 BETH + 1,584.30217811 USDT

This calculation shows an estimated impermanent loss. This is provided as a reference only and it does not represent actual impermanent loss you can incur. Please refer to our FAQs to learn more about impermanent loss.

While liquidity providers can use stablecoins, yields, and rewards to help lessen the impact of impermanent loss they can also reduce this by using liquidity pools that use ratios other than 50/50. Balancer is a platform that offers liquidity pools with ratios like 60/40 or 80/20. When ETH is deposited into a pool that is 50/50 the liquidity provider has to have 50% exposure to another token. With an 80/20 pool, they only need 20% exposure to another token. You can see below how three liquidity pool ratios are affected by impermanent loss differently, with the 95/5 pool seeing the least impermanent loss.

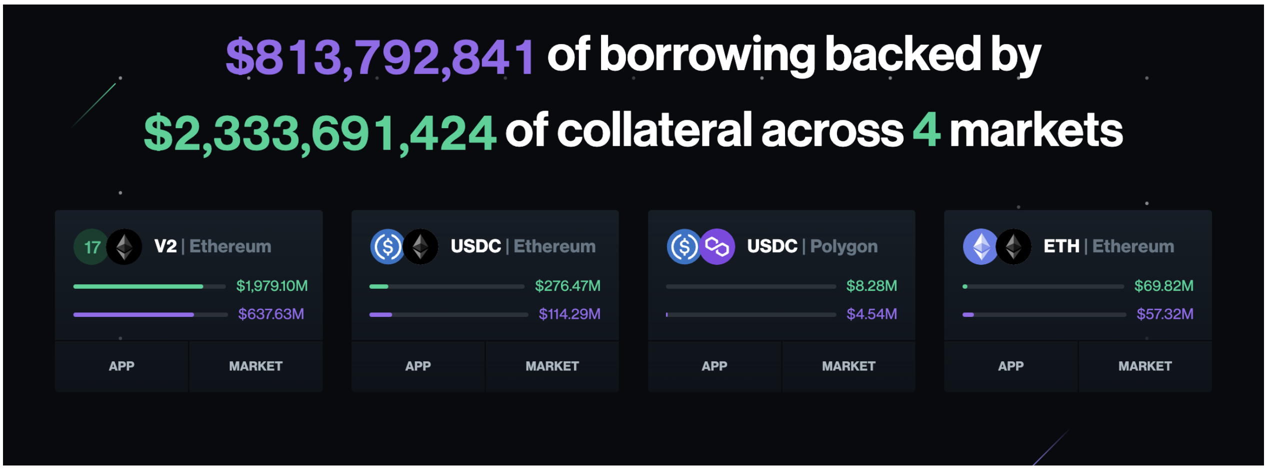


Borrowing / Lending

Compound

Compound III is an EVM compatible protocol that enables supplying of crypto assets as collateral in order to borrow the *base asset*. Accounts can also earn interest by supplying the base asset to the protocol.

The initial deployment of Compound III is on Ethereum and the base asset is USDC.



$813,792,841 of borrowing backed by

$2,333,691,424 of collateral across 4 markets

V2 | Ethereum
$1,979.10M
$637.63M

USDC | Ethereum
$276.47M
$114.29M

USDC | Polygon
$8.28M
$4.54M

ETH | Ethereum
$69.82M
$57.32M

Interacting with Uniswap

Guide to single swaps

From Uniswap docs

Swaps are the most common interaction with the Uniswap protocol. The `exactInputSingle` function is for performing *exact input* swaps, which swap a fixed amount of one token for a maximum possible amount of another token. This function uses the `ExactInputSingleParams` struct and the `exactInputSingle` function from the ISwapRouter interface.

When trading from a smart contract, the most important thing to keep in mind is that access to an external price source is required. Without this, trades can be front run for considerable loss.

Exact Input Swaps

The caller must `approve` the contract to withdraw the tokens from the calling address's account to execute a swap. Remember that because our contract is a contract itself and not an extension of the caller (us); we must also approve the Uniswap protocol router contract to use the tokens that our contract will be in possession of after they have been withdrawn from the calling address (us).

To execute the swap function, we need to populate the `ExactInputSingleParams` with the necessary swap data. These parameters are found in the smart contract interfaces, which can be browsed here.

The function parameters:

- `tokenIn` The contract address of the inbound token
- `tokenOut` The contract address of the outbound token
- `fee` The fee tier of the pool, used to determine the correct pool contract in which to execute the swap
- `recipient` the destination address of the outbound token
- `deadline`: the unix time after which a swap will fail, to protect against long-pending transactions and wild swings in prices

- `amountOutMinimum`: we are setting to zero, but this is a significant risk in production. For a real deployment, this value should be calculated using our SDK or an onchain price oracle - this helps protect against getting an unusually bad price for a trade due to a front running sandwich or another type of price manipulation
- `sqrtPriceLimitX96`: We set this to zero - which makes this parameter inactive. In production, this value can be used to set the limit for the price the swap will push the pool to, which can help protect against price impact or for setting up logic in a variety of price-relevant mechanisms.

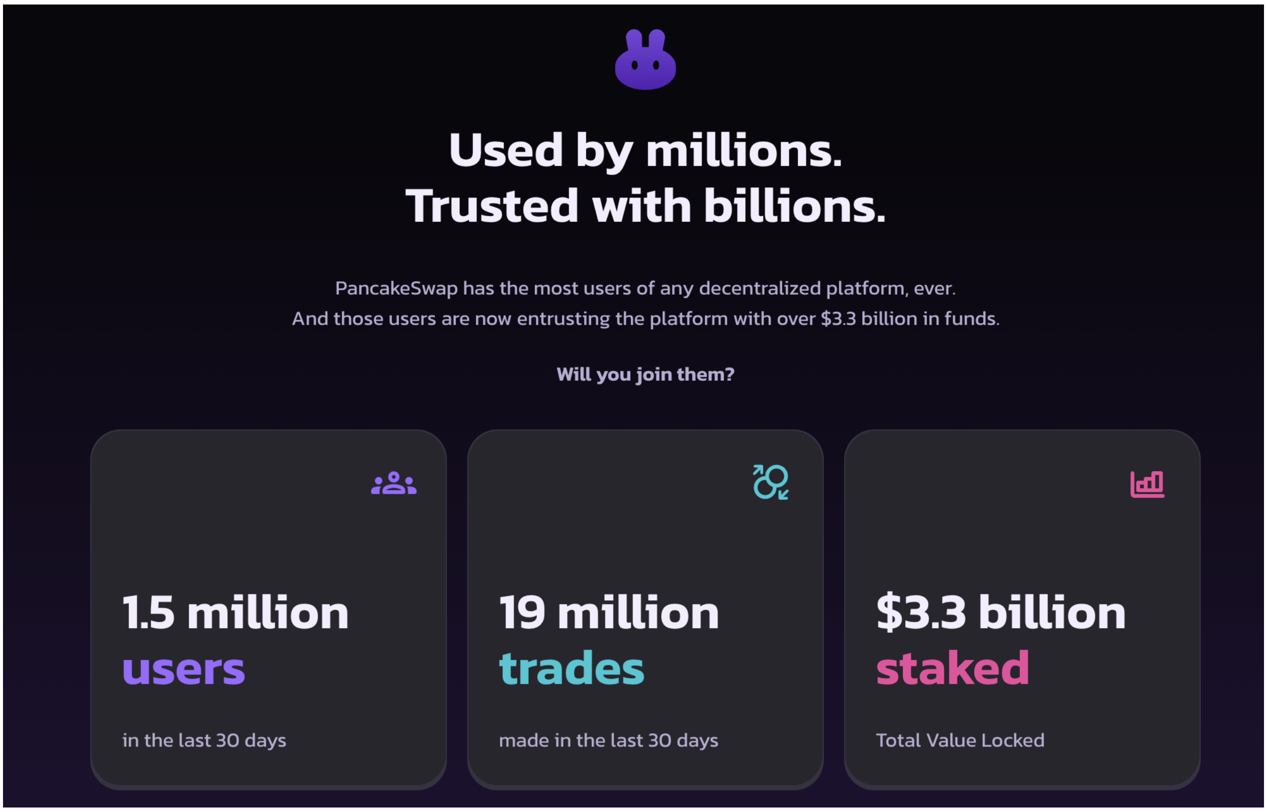
Calling the function from Solidity

```
// Naively set amountOutMinimum to 0. In production, use an
oracle or other data source to choose a safer value for
amountOutMinimum.
// We also set the sqrtPriceLimitx96 to be 0 to ensure we swap
our exact input amount.

                ISwapRouter.ExactInputSingleParams memory
params =
            ISwapRouter.ExactInputSingleParams({
                tokenIn: DAI,
                tokenOut: WETH9,
                fee: poolFee,
                recipient: msg.sender,
                deadline: block.timestamp,
                amountIn: amountIn,
                amountOutMinimum: 0,
                sqrtPriceLimitX96: 0
            });

        // The call to `exactInputSingle` executes the swap.
        amountOut = swapRouter.exactInputSingle(params);
    }
```

Pancake Swap



Used by millions.
Trusted with billions.

PancakeSwap has the most users of any decentralized platform, ever.
And those users are now entrusting the platform with over $3.3 billion in funds.

Will you join them?

1.5 million users

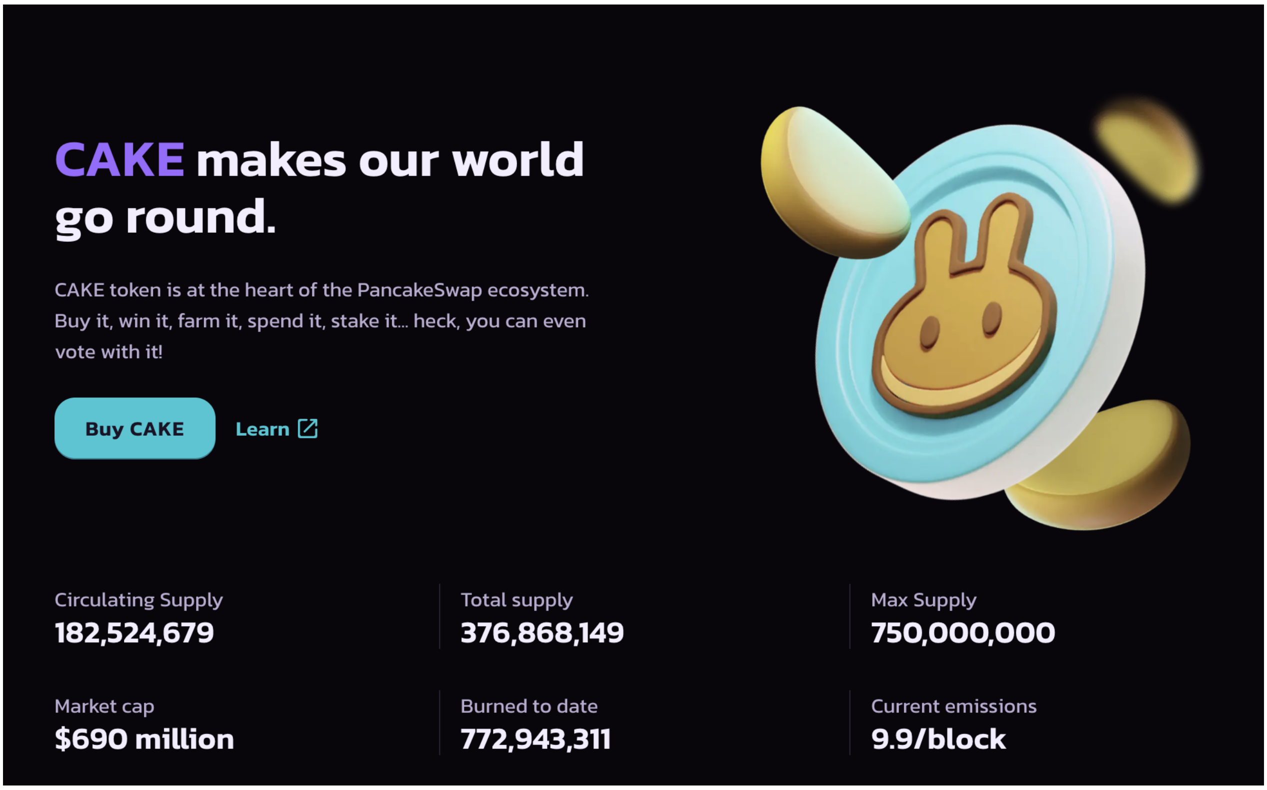
in the last 30 days

19 million trades

made in the last 30 days

$3.3 billion staked

Total Value Locked

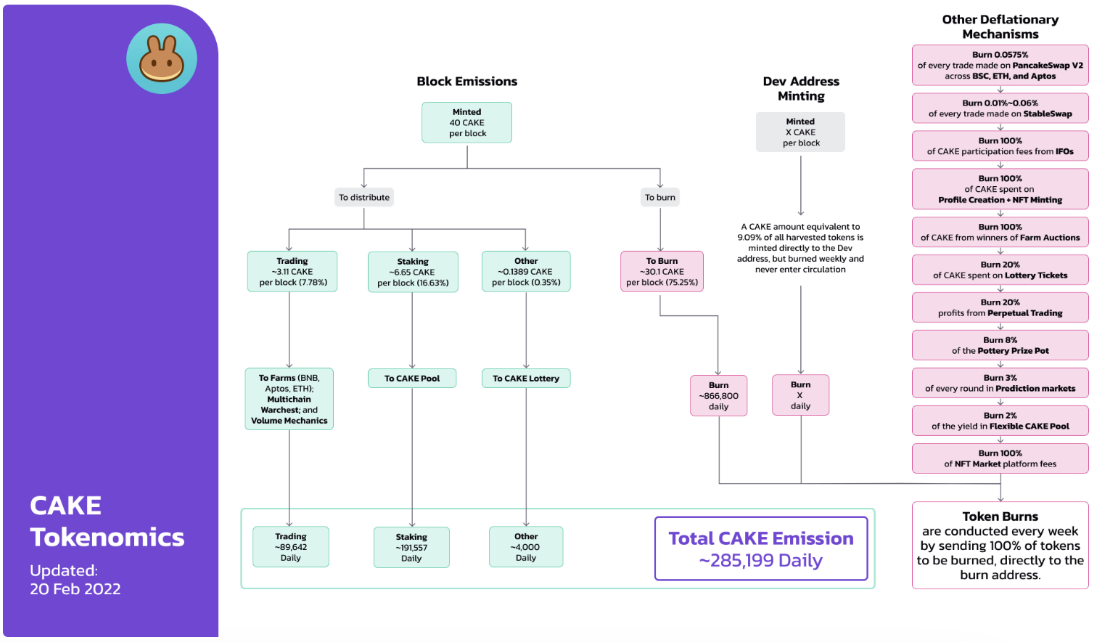


CAKE makes our world go round.

CAKE token is at the heart of the PancakeSwap ecosystem. Buy it, win it, farm it, spend it, stake it... heck, you can even vote with it!

Circulating Supply **182,524,679**	Total supply **376,868,149**	Max Supply **750,000,000**
Market cap **$690 million**	Burned to date **772,943,311**	Current emissions **9.9/block**

CAKE Tokenomics

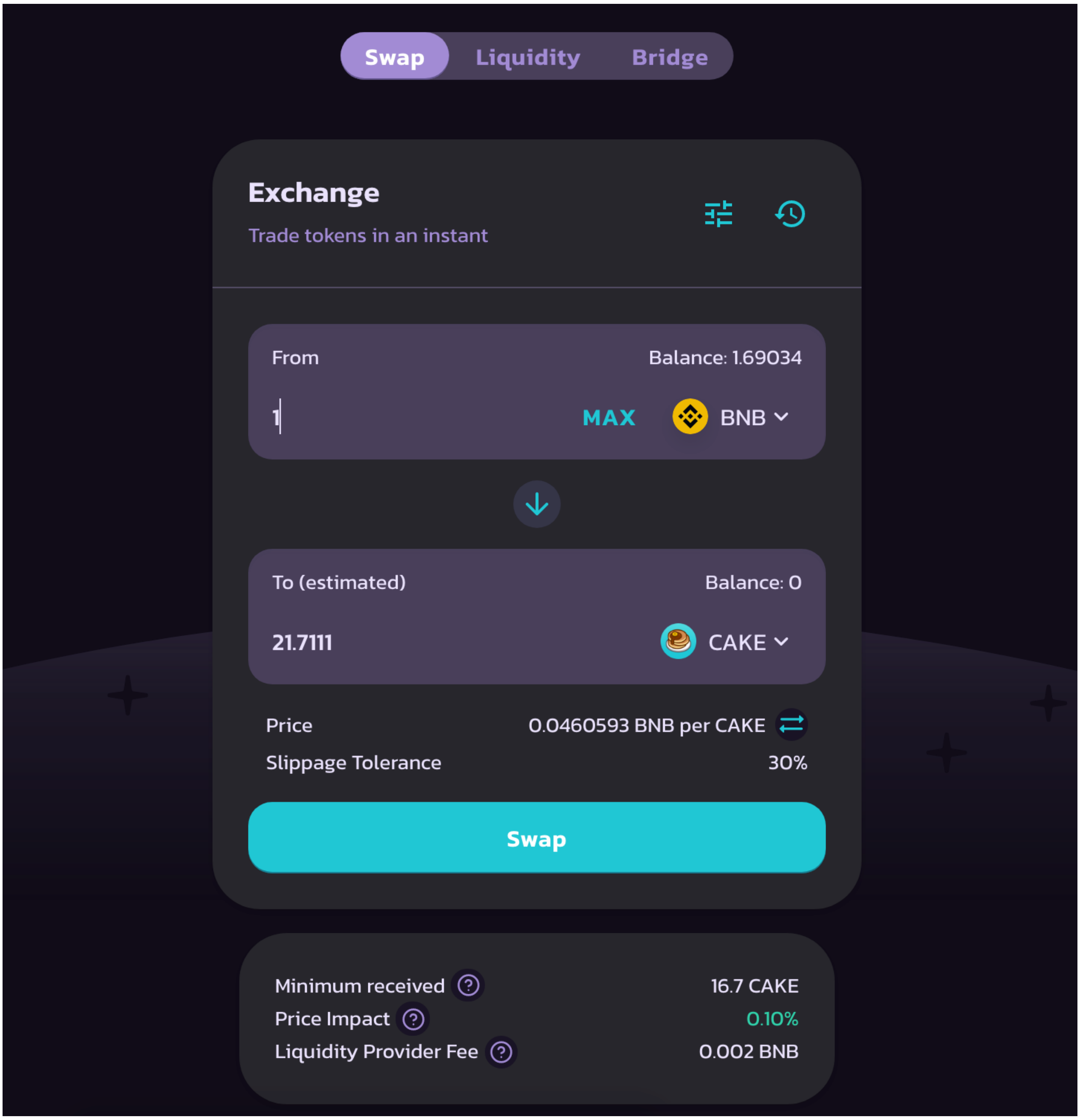

CAKE Tokenomics
Updated:
20 Feb 2022
Block Emissions
Minted 40 CAKE per block
To distribute
To burn
Trading ~3.11 CAKE per block (7.78%)
Staking ~6.65 CAKE per block (16.63%)
Other ~0.1389 CAKE per block (0.35%)
To Burn ~30.1 CAKE per block (75.25%)
To Farms (BNB, Aptos, ETH); Multichain Warchest; and Volume Mechanics
To CAKE Pool
To CAKE Lottery
Burn ~866,800 daily
Dev Address Minting
Minted X CAKE per block
A CAKE amount equivalent to 9.09% of all harvested tokens is minted directly to the Dev address, but burned weekly and never enter circulation
Burn X daily
Trading ~89,642 Daily
Staking ~191,557 Daily
Other ~4,000 Daily
Total CAKE Emission ~285,199 Daily
Other Deflationary Mechanisms
Burn 0.0575% of every trade made on PancakeSwap V2 across BSC, ETH, and Aptos
Burn 0.01%~0.06% of every trade made on StableSwap
Burn 100% of CAKE participation fees from IFOs
Burn 100% of CAKE spent on Profile Creation + NFT Minting
Burn 100% of CAKE from winners of Farm Auctions
Burn 20% of CAKE spent on Lottery Tickets
Burn 20% profits from Perpetual Trading
Burn 8% of the Pottery Prize Pot
Burn 3% of every round in Prediction markets
Burn 2% of the yield in Flexible CAKE Pool
Burn 100% of NFT Market platform fees
Token Burns are conducted every week by sending 100% of tokens to be burned, directly to the burn address.

Adding Liquidity

Guide to adding / removing liquidity

Swapping Tokens

Token swaps on PancakeSwap are a simple way to trade one BEP-20 token for another via automated liquidity pools, and also with market makers when trading ERC-20 tokens on Ethereum.



Exchange

Trade tokens in an instant

From Balance: 1.69034

1 BNB

To (estimated) Balance: 0

21.7111 CAKE

Price 0.0460593 BNB per CAKE

Slippage Tolerance 30%

Minimum received 16.7 CAKE

Price Impact 0.10%

Liquidity Provider Fee 0.002 BNB

See guide

Smart Router

PancakeSwap Smart Router is a routing algorithm that links the AMM and stableswap (BNB Chain), and the AMM and market makers (Ethereum), to provide better liquidity and pricing. It uses a smart order routing algorithm that executes trades across multiple pools to find the best price for traders. For more information on StableSwap click here and for the Market Maker integration click here.

Interacting with Pancake Swap

Important Contracts

Factory

View PancakeFactory.sol on GitHub.

Binance smart chain Contract address
0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73
View the PancakeSwap: Factory v2 contract on BscScan

The Factory is the contract that holds details of the token pairs.

Interface

pragma solidity =0.5.16;

```
interface IPancakeFactory {
    event PairCreated(address indexed token0, address indexed
token1, address pair, uint);
    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);
    function getPair(address tokenA, address tokenB) external
view returns (address pair);
    function allPairs(uint) external view returns (address
pair);
    function allPairsLength() external view returns (uint);
    function createPair(address tokenA, address tokenB)
external returns (address pair);

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;
}
```

getPair

```
function getPair(address tokenA, address tokenB) external view
returns (address pair);
```

Address for `tokenA` and address for `tokenB` return address of pair contract (where one exists).
`tokenA` and `tokenB` order is interchangeable.
Returns `0x0000000000000000000000000000000000000000` as address where no pair exists.

createPair

```
function createPair(address tokenA, address tokenB) external
returns (address pair);
```

Creates a pair for `tokenA` and `tokenB` where a pair doesn't already exist.
`tokenA` and `tokenB` order is interchangeable.
Emits `PairCreated`

Router

Contract name: PancakeRouter

View PancakeRouter.sol on GitHub.

Binance smart chain Contract address
0×10ED43C718714eb63d5aA57B78B54704E256024E
View the PancakeSwap: Router v2 contract on BscScan

Interface
https://gist.github.com/extropyCoder/fad53a96998912acfe6246f47209f4e0

Add Liquidity

```
function addLiquidity(
  address tokenA,
  address tokenB,
  uint amountADesired,
  uint amountBDesired,
  uint amountAMin,
  uint amountBMin,
  address to,
  uint deadline
) external returns (uint amountA, uint amountB, uint
liquidity);
```

```
function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
) external returns (uint[] memory amounts);
```

DeFi Development

Making a fork of mainnet

Hardhat

See hardhat documentation

You first need to have an account on Quick Node
This will give you a key so that you can use their RPC nodes.

Forking using ganache

`npx ganache-cli --f https://<node details> -m “your 12 word mnemonic” --unlock <address> -i <chain ID>`

Fork from hardhat

```
npx hardhat node --fork https://<node details>
```

In hardhat you can also specify this in the config file

```
networks: {
  hardhat: {
    forking: {
      url: "https://<node details>",
    }
  }
}
```

Foundry

You can use:

`forge test --fork-url <https://<node details>> -vv`

where your_rpc_url is your node details as above.

Or you can use Anvil (part of the Foundry suite) for more control:

> `anvil --fork-url <https://<node details>>`
>
> With anvil you are able to use the `anvil_impersonateAccount` custom method to impersonate EOA's.
>
> Please read here for more info