What Is Sudoswap?

The sudoswap AMM – or just *sudoswap* for short – is a minimal, gas-efficient automated market maker (AMM) protocol that facilitates NFT-to-token swaps (and vice versa) using customizable bonding curves. sudoswap supports ERC721 NFTs, as well as all ETH and ERC20 tokens.

Liquidity providers (LPs) can deposit assets into single-sided buy or sell pools, or into dual-sided trade pools which buy *and* sell NFTs with an optional spread to capture trading fees.

Similar to other floor NFT protocols, sudoswap makes no distinction between different ERC721 IDs. Pools that are willing to buy or sell NFTs will return the same price no matter which NFT is sent in or out from the collection.

How Does Sudoswap Work?

Sudoswap is an AMM protocol for NFTs, which means that users buy from or sell into liquidity pools instead of directly trading between themselves. If you're familiar with Uniswap, it's a similar concept but for NFTs.

Here's how it works: 1. Liquidity providers deposit NFTs and/or ETH (or an ERC20 token) into liquidity pools. They choose whether they would like to buy or sell NFTs (or both) and specify a starting price and bonding curve parameters. 2. Users can then buy NFTs from or sell NFTs into these pools. Every time an item is bought or sold, the price to buy or sell another item changes for the pool based on its bonding curve. 3. At any time, liquidity providers can change the parameters of their pool or withdraw assets.

What Is A Liquidity Pool?

A pool, or liquidity pool, is a smart contract that allows you to instantly swap between two assets. On sudoswap, the most common type of pool is an NFT<>ETH pool, which means that anyone holding NFTs from that collection can instantly swap them for ETH, or vice versa.

Pools use a bonding curve to determine the relative price at which one asset is traded for another. The more an asset is bought from the pool, the more expensive it becomes. Conversely, the more an asset is sold to the pool, the cheaper it becomes.

Ideally, a pool contains some amount of both assets, enabling users to swap back and forth between them. However, it's also possible to create a pool with just one asset, meaning that users will only be able to buy that asset from the pool.

A bonding curve is a mathematical formula which defines the relationship between an asset's price and its supply. Bonding curves are a key feature of automated market makers since they are used to algorithmically adjust asset prices.

sudoswap supports three types of bonding curve: linear, exponential, and XYK (constant product).

Linear

With a linear bonding curve, the price of an NFT is increased by a flat amount (called `delta`) every time an item is bought from the pool. Conversely, the price of the NFT is decreased by that same flat amount every time an item is sold to the pool.

For example, a liquidity provider may create an NFT<>ETH pool with a `Start Price` of 1 ETH and a `delta` of 0.1 ETH. Assuming they provide enough liquidity, the price of an NFT will increase to 1.1 ETH after one item is purchased from the pool. After a second item is purchased, the price will increase to 1.2 ETH, and so on and so forth. At any point, if an NFT is sold to the pool, the price will decrease by 0.1 ETH.

Exponential

With an exponential bonding curve, the price of an NFT is increased by a certain percentage (also called `delta`) every time an item is bought from the pool. Conversely, the price of the NFT is decreased equivalently every time an item is sold to the pool.

To calculate the equivalent decrease, convert the percentage to a decimal index (e.g. for 50%, the index would be 1.5) and divide the price by this number.

For example, a liquidity provider may create an NFT<>ETH pool with a `Start Price` of 2 ETH and a `delta` of 50%. Assuming they provide enough liquidity, the price of an NFT will increase to 2 + 50% = 3 ETH after one item is purchased from the pool. After a second item is purchased, the price will increase to 3 + 50% = 4.5 ETH, and so on and so forth. At any point, if an NFT is sold to the pool, the price will be divided by 1.5.

XYK (Constant Product)

With an XYK curve, the price of an NFT is adjusted every time an item is bought from or sold to the pool, such that the product of two virtual reserves remains constant after every trade. These virtual reserves correspond to the number and value of NFTs the pool will buy or sell.

An additional concentration parameter allows liquidity providers to adjust (i.e. tighten or loosen) XYK curves.

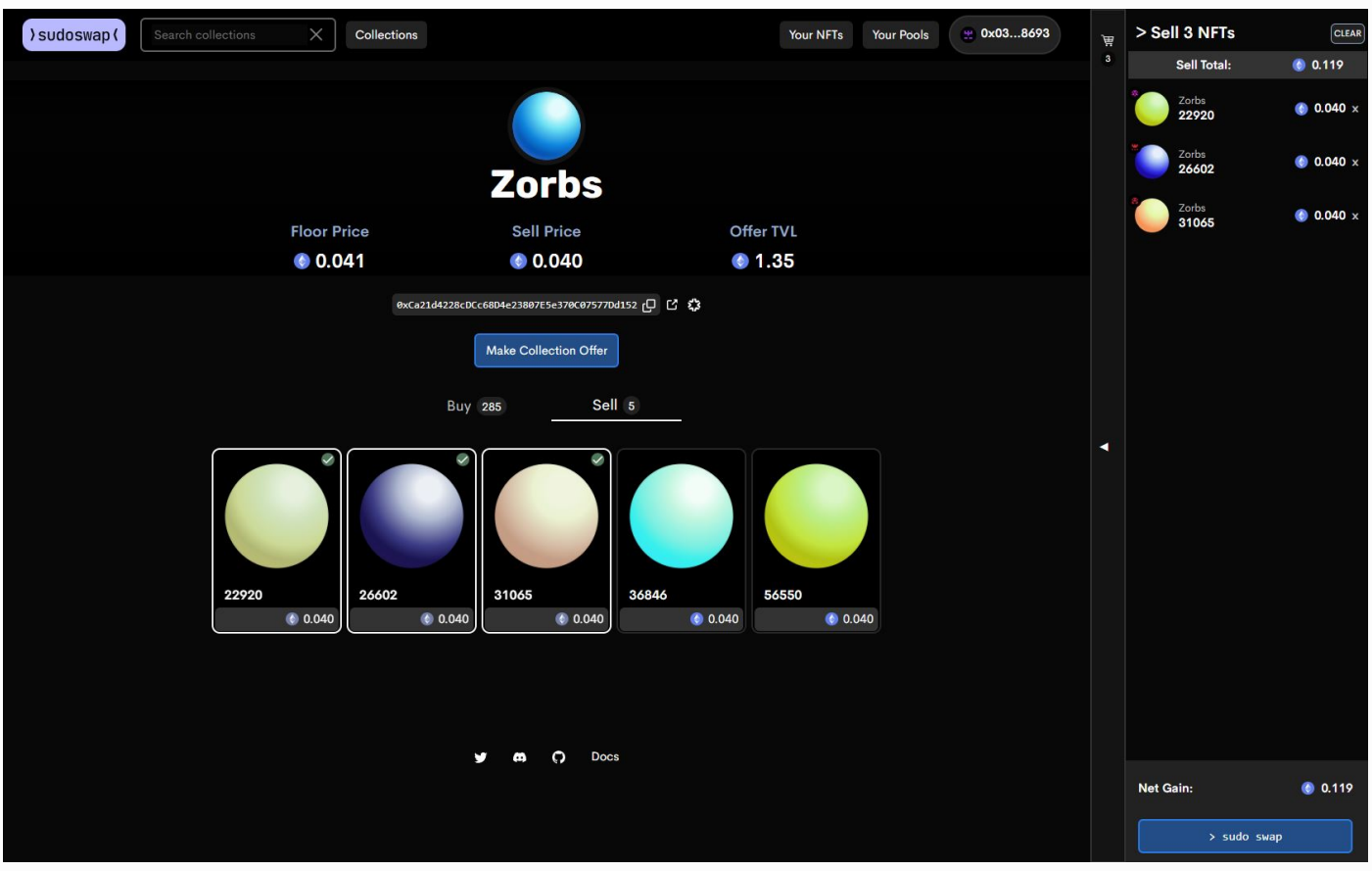
For information on how exactly pricing is calculated for XYK curves, refer to the Pricing page in the technical reference.

Next

1. Click "> sudo swap" to initiate the swap.
2. Confirm the transaction in your wallet.

Selling NFTs

1. Go to the Collections page and select a collection.
2. Open the Sell tab at the center of the page:



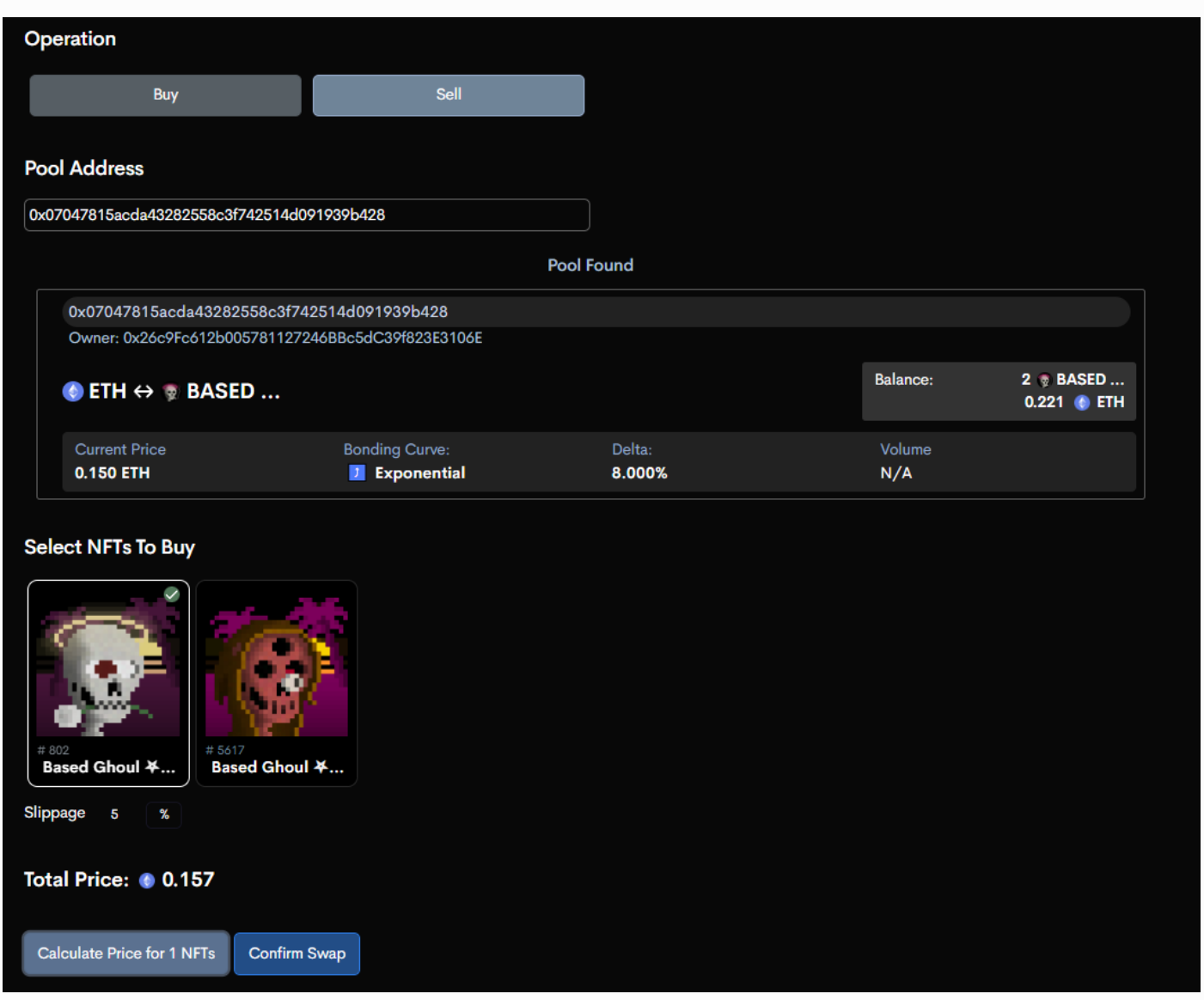
1. Select the NFTs you want to sell and click "> sudo swap".
2. Give sudoswap access to the NFTs by confirming the first transaction in your wallet.
3. Finalize the sale by confirming the second transaction in your wallet.

Direct Pool Swap

When buying or selling NFTs using the steps above, sudoswap will attempt to automatically route your order to the liquidity pool with the best price.

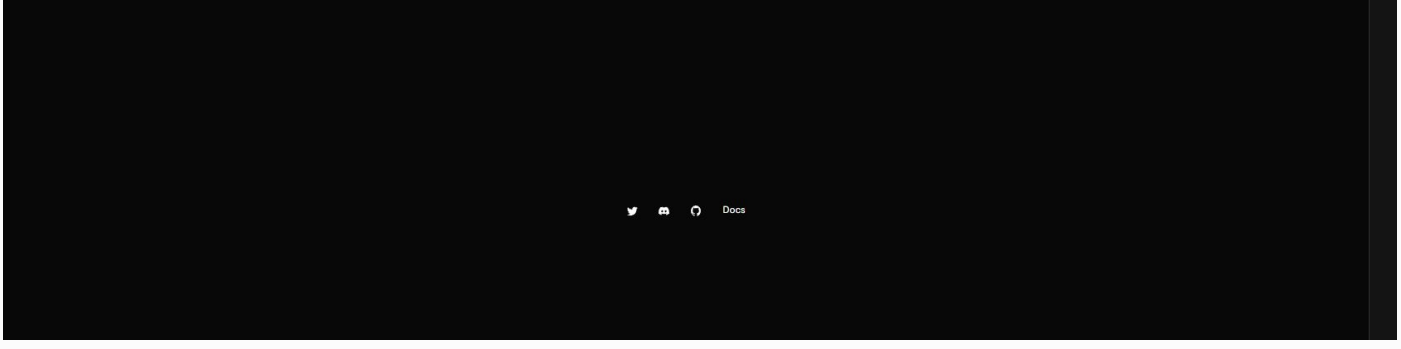
In times of high demand, you may wish to buy or sell from a specific liquidity pool while specifying a maximum slippage tolerance. This is called a *Direct Pool Swap*.

1. Go to the Collections page and select a collection.
2. Open the Pools tab to see a list of liquidity pools.
3. Click on the pool with the best price for your buy or sell, making sure its ETH/NFT balance is sufficient.
4. Select Direct Pool Swap at the top-left of the pool page.
5. Click on Buy or Sell and choose which NFTs you would like to buy/sell:

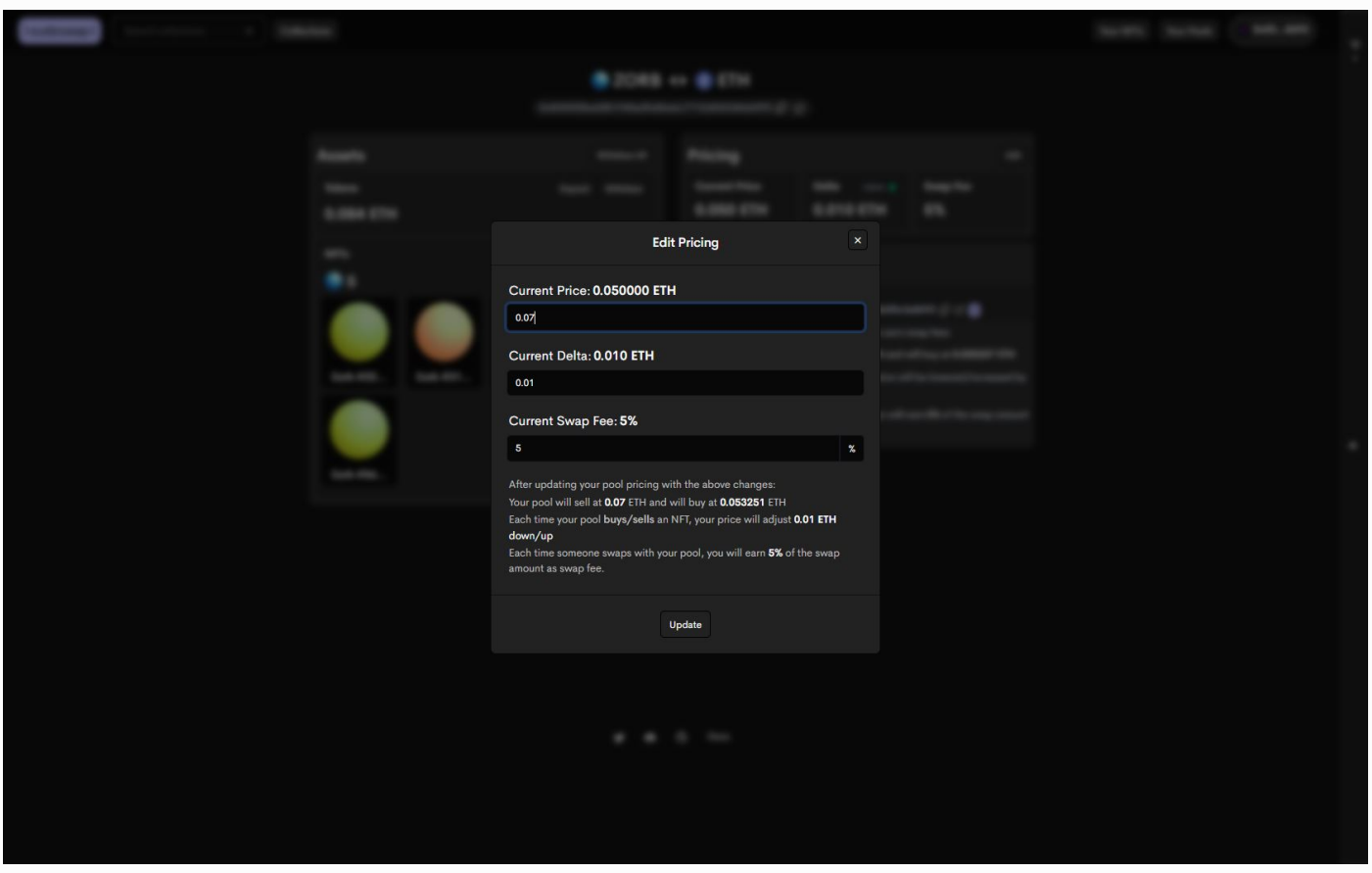


1. Enter a slippage tolerance, which is the maximum price change you are willing to accept if the price of the pool changes before your transaction is confirmed. **To prevent frontrunning, avoid setting a slippage tolerance higher than the pool's delta.**
2. Click on "Calculate Price for NFTs".
3. Once the total price loads, click on "Confirm Swap".
4. Finalize the swap by confirming the transaction(s) in your wallet.

Previous Next



1. Enter a new start price and delta for the pool:

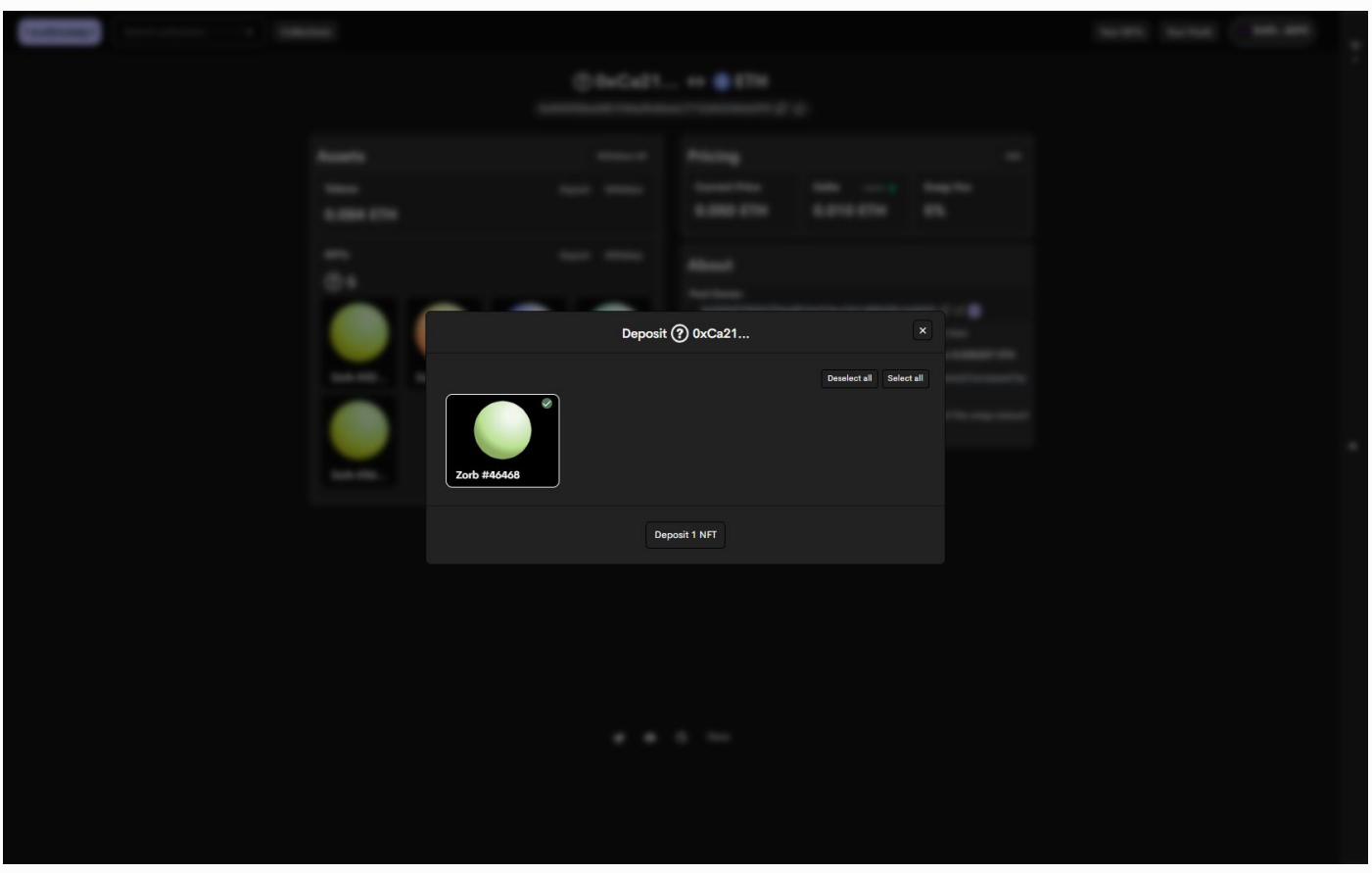


1. Click "Update" and confirm the transaction in your wallet.

Note: At this time it is not possible to convert a linear pool into an exponential pool or vice versa. To do this, you must withdraw your assets from the existing pool and create a new one.

Depositing NFTs

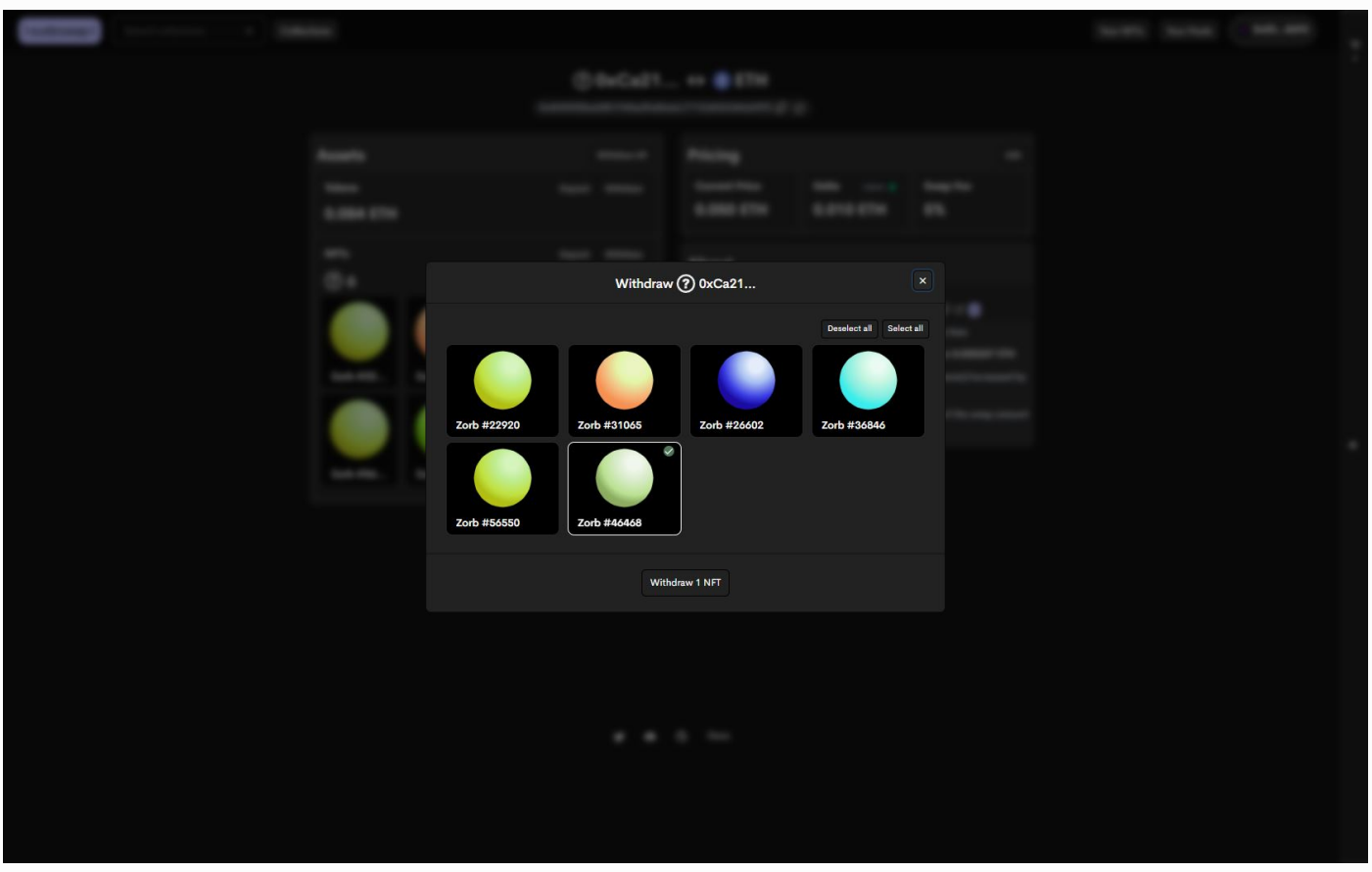
1. Navigate to the Your Pools tab at the top-right of the page.
2. Click on the pool you want to deposit to.
3. Click on the "Deposit" button at the top-left of the pool.
4. Select the NFTs you want to deposit:



1. Click "Deposit NFTs" and confirm the transaction in your wallet.

Withdrawing NFTs

1. Navigate to the Your Pools tab at the top-right of the page.
2. Click on the pool you want to withdrawm from.
3. Click on the "Withdraw" button at the top-left of the pool.
4. Select the NFTs you want to withdraw:



1. Click "Withdraw NFTs" and confirm the transaction in your wallet.

Previous

Next

Frequently Asked Questions

This page contains some of the most frequently asked questions among sudoswap users.

Can I withdraw airdropped tokens from my pool?

Yes. At any time, you can withdraw any ERC20, ERC721, or ERC1155 tokens from your liqudity pools using the "Withdraw Other Tokens" feature on the pool's page. Note that you can only withdraw tokens from pools you have created.

What's the difference between selling and listing?

When you *sell* an NFT on sudoswap, it is immediately sold into a bonding curve for the best price possible. This means that you receive the proceeds of the sale right away.

When you *list* an NFT on sudoswap, you can set whatever price you want, but you have to wait until someone buys it from you.

Do I get a token representing my liquidity position?

No, you will not receive a token representing your liquidity position when creating a new pool. This is because every liquidity pool has a unique smart contract associated with and owned by its creator.

All assets pertaining to a pool are held in the pool's smart contract, so there is no need for additional tokens to keep track of deposits.

Why is my new liquidity pool not showing up on the sudoswap website?

Sudoswap indexes on-chain data at regular intervals. If a new liquidity pool is not showing up, it is usually because sudoswap has not yet had a chance to index the pool's creation. Please check again at a later time.

What happens to a trade pool when one side of the pool is depleted?

Trade pools remain trade pools even if one side of the pool is depleted. Nevertheless, pools can only fulfill buys and sells if they hold the necessary assets to do so.

Listing on Sudoswap

Compatible ERC721 NFTs can be deposited and traded on sudoswap immediately after they are minted, without the need for collections to be manually listed.

For a collection to show up in the search bar, it must have at least one liquidity pool or past volume on sudoswap. However, even if a collection does not show up, holders can still create liquidity pools for it as normal.

Launching on Sudoswap

There is no special procedure to launch an NFT collection on sudoswap. However, if you'd like to make sudoswap part of your launch, you can: 1. Deploy an NFT smart contract as you would for any other collection. 2. Mint part or all of the supply to an address you control. 3. Create a liquidity pool on sudoswap with those NFTs. 4. Tell collectors to buy from the liquidity pool.

When launching a collection like this, it is advised to deposit no more than a few hundred NFTs in each liquidity pool. Depositing too many NFTs in a single pool may cause poor performance for collectors trading through the sudoswap website.

Frequently Asked Questions

This page contains some of the most frequently asked questions among NFT collection owners.

Why isn't my collection showing in the search bar?

For a collection to show up in the search bar, it must have at least one liquidity pool or past volume on sudoswap. If your collection only recently met this requirement, please allow some time for sudoswap to process this.

Previous Next

Creating A Pair

New pairs for the sudoswap AMM are created with the `LSSVMPairFactory`. LPs will call either `createPairETH` or `createPairERC20` depending on their token type (i.e. if they wish to utilize ETH or an ERC20). This will deploy a new `LSSVMPair` contract.

Each pair has one `owner` (initially set to be the caller), and multiple pools for the same token and NFT pair can exist, even for the same owner. This is due to each pair having its own potentially different spot price and bonding curve.

Pair Types

Each pair can be one of 3 types: Token, NFT, or Trade.

A Token pair holds either ETH or an ERC20 token and will give a quote for how much it will pay for any NFT in the collection it has been created for.

An NFT pair holds NFTs and will give a quote for how much it will sell for any NFT it has in its inventory.

A Trade pair holds both NFTs and tokens, and it will give a quote for both buying and selling, with a spread between the quotes as the LP fee.

Pair Parameters

During the creation call, LPs supply the following parameters:

```
IERC721 _nft, // The ERC721 token side of the pair
ICurve _bondingCurve, // The bonding curve used to price the pair
address payable _assetRecipient, // If set to a non-zero address, assets sent to the pair durin
LSSVMPair.PoolType _poolType, // The pair's type (Token, NFT, Trade)
uint256 _delta, // The bonding curve's parameter (more on this in the Bonding Curve section)
uint256 _fee, // The spread between buy and sell quotes. NOTE: only for Trade pairs.
uint256 _spotPrice, // The initial sell quote the pair will return if an NFT is sold into it.
uint256[] calldata _initialNFTIDs // The list of IDs to transfer from the caller to the pair
```

For an ERC20 pair, two other parameters are also added:

```
ERC20 token // The token to be used for the token side of the pair.
uint256 initialTokenBalance // The amount of tokens to send into the pair. (The createPairETH c
```

The factory uses a modified minimal proxy pattern to allow deployed pairs to cheaply access certain values the same way that `immutable` normally works. See `LSSVMPairCloner` for more technical details.

As a result, several of the above parameters *cannot* be changed after a pool is deployed:

```
_nft,
_bondingCurve,
_poolType,
token // NOTE: only for ERC20 pairs
```

Managing A Pair

The `owner` of a pair can modify certain parameters to change the pair's pricing and inventory.

To modify `spotPrice`, `delta`, and `fee`, the `owner` can call `changeSpotPrice`, `changeDelta`, and `changeFee` (only for Trade pairs).

To withdraw any NFTs or tokens sent to the pair, the `owner` can call `withdrawERC721`, `withdrawERC20`, and `withdrawETH` (only for ETH pairs).

Previous Next

`LSSVMPair` contract itself.

Linear Curve

The linear curve performs an additive operation to update the price. `delta` is assumed to be set properly by the LP to be the same precision of the pair's underlying token.

If the pair has just sold an NFT by giving out an NFT and receiving tokens, the next price it will quote to sell NFTs at will be `delta` more. Conversely, if the pair has just bought an NFT by giving out tokens and receiving an NFT, the next price it will quote to purchase NFTs at will be `delta` less.

Exponential Curve

The exponential curve performs a multiplicative operation. `delta` is treated as a multiplier assuming a fixed point system where `1e18` is 1.

For example, if `delta` is `1e18 + 1e17`, this represents a 10% change in price each time.

If the pair has just sold an NFT by giving out an NFT and receiving tokens, the next price it will quote to sell NFTs at will be multiplicatively `delta` more. Conversely, if the pair has just bought an NFT by giving out tokens and receiving an NFT, the next price it will quote to purchase NFTs at will be multiplicatively `delta` less.

XYK Curve

The XYK curve adjusts price such that the product (multiplication) of two virtual reserves remains constant after every trade. At pool creation,the virtual reserves are set as follows:

- `nftBalance`: the number of NFTs being bought or sold (in the case of trade pools, whichever is greater) **plus one**
- `tokenBalance`: the number of NFTs being bought or sold (or whichever is greater) multiplied by the start price

To conform to the `IPair` interface, `nftBalance` is stored as `delta` and `tokenBalance` is stored as `spotPrice`.

The net price (exclusive of pool and protocol fees) for an XYK pair to sell *X* NFTs is `inputValueWithoutFee = (x * tokenBalance) / (nftBalance - x)`. Conversely, the net price for an XYK pair to buy *X* NFTs is `outputValueWithoutFee = (x * tokenBalance) / (nftBalance + x)`.

Immediately after every trade, the virtual reserves are updated:

- Pair sells: `nftBalance = nftBalance - x` and `tokenBalance = tokenBalance + outputValueWithoutFee`
- Pair buys: `nftBalance = nftBalance + x` and `tokenBalance = tokenBalance - outputValueWithoutFee`

Understanding Spot Price

In addition to modifying `delta` to change the pair's price reactivity, it is important to understand how the `spotPrice` variable in `LSSVMPair` behaves.

The spot price refers to the instantaneous price of *selling* 1 NFT *to* the pair. The instantaneous price of *buying* 1 NFT *from* the pair is set to be the `spotPrice` adjusted upwards by 1 unit of `delta` with respect to the pair's bonding curve.

For example, say that we have a Trade `LSSVMPair` for ETH with a `spotPrice` of 1 ETH, and a linear curve with a `delta` of 0.1 ETH. (Assume `fee` is 0.) Then a user selling 1 NFT to the pair would receive 1 ETH, whereas a user purchasing 1 NFT from the pair would have to send (1 + 0.1) = 1.1 ETH.

In other words, the price to purchase an NFT from a pair will always be `delta` greater (either additively on multiplicatively) than the price to sell an NFT to the pair.

Pair managers who are manually adjusting `spotPrice` should keep this in mind.

(Note that for simplicity, some examples in this documentation may use "spot price" to also refer to the instantaneous buy price.)

Pricing For Multi-Swaps

If a user buys or sells multiple NFTs in one swap transaction, the `spotPrice` will update by `delta` for each NFT bought or sold.

For example, say that we have a Trade `LSSVMPair` for ETH with a `spotPrice` of 1 ETH, and a linear curve with a `delta` of 0.1 ETH. (Assume `fee` is 0.)

If a user sells 5 NFTs to this pair, they will receive:

- 1 ETH for the first NFT
- 0.9 ETH for the second NFT
- 0.8 ETH for the third NFT
- 0.7 ETH for the fourth NFT
- 0.6 ETH for the fifth NFT

At the end of the multi-swap, the new `spotPrice` will be set to 0.5 ETH.

Swapping Across Pairs

The recommended way to make swaps across various pairs is to use the `LSSVMRouter`. Users can set allowances for tokens and NFTs once for the router, instead of for each new pool they wish to swap with.

On the protocol level, the sudoswap AMM does not perform any routing optimization on-chain. Users are expected to know their desired swap paths when calling the router, e.g. via the use of an off-chain indexing service.

NFT<>Token Swaps

When swapping tokens for NFTs, users can either specify which NFT IDs they want from each pair, or they can ask for any ID from the pair.

When swapping from NFT to tokens or from tokens to NFTs, the `LSSVMRouter` has two types of swaps: a Normal Swap and a Robust Swap.

Normal Swap

A Normal Swap is conceptually similar to token-to-token swaps on other DEXs. The user sends the router a maximum input amount or minimum output amount (i.e. allowed slippage), as well as a swap route and a deadline. The router will then swap across the various pairs specified. At the end of all the swaps, the router will total all tokens to be received or sent, and revert if the total is beyond the user's specified slippage.

Robust Swap

In contrast, a Robust Swap does a slippage check *per* swap pair rather than an *aggregate* check at the end. If the price for a specified swap pair is past the allowable slippage, the router will silently skip that route and move on to the next one, with no reverts or errors.

This is intended for situations where the price can move quickly between your transaction submission and execution.

Normal vs Robust Swap

To see the difference between these swap types, consider this example: say there are 2 pairs for NFT and ETH. The first pair has a spot price of 1 ETH and a linear curve with a delta of 0.1 ETH. The second pair has a spot price of 1 ETH and a linear curve with a delta of 1 ETH.

A user wishes to purchase NFTs, one from each pool for 1 ETH each, with 10% slippage.

The user submits a swap transaction. Before this transaction is executed, someone else goes and buys 1 NFT from each pool for 1 ETH each.

The new spot price for the first pair is 1.1 ETH, and for the second pair is 2 ETH.

If the user had submitted a Normal Swap, they would have sent 2.2 ETH (2 ETH + 0.2 ETH to cover the additional 10% slippage), and the transaction would fail, as they would have sent enough ETH to cover the first swap at the new price of 1.1 ETH, but not enough to cover the second swap at 2 ETH.

In contrast, if the user had submitted a Robust Swap, they would have also sent 2.2 ETH, but with a *per-swap* max cost of 1.1 ETH. The router would have enough to cover the first swap at 1.1 ETH. Then, upon seeing the second swap would cost 2 ETH, the router would skip this pair entirely. The transaction would succeed, refunding 1.1 ETH back to the user, as well as sending them the one NFT they purchased for 1.1 ETH.

Thus, the Robust Swap is generally recommended for a better user experience in higher volatility environments, at the cost of slightly more gas.

NFT<>NFT Swaps

As another added convenience, the `LSSVMRouter` also supports swapping NFTs for tokens, and then tokens to other NFTs in one transaction.

Like the token to NFT swap, users can either specify specific NFT IDs from a pair, or ask for any NFT ID.

Previous Next

Goerli Contract Addresses

EXPONENTIAL_CURVE: 0x0D807bd5fF2C4eF298755bE30E22926b33244B0c

LINEAR_CURVE: 0xaC6dcFF6E13132f075e36cA3a7F403236f869438

XYK_CURVE: 0x02363a2F1B2c2C5815cb6893Aa27861BE0c4F760

PAIR_FACTORY: 0xF0202E9267930aE942F0667dC6d805057328F6dC

PAIR_ROUTER: 0x25b4EfC43c9dCAe134233CD577fFca7CfAd6748F

Mainnet Contract Addresses

EXPONENTIAL_CURVE: 0x432f962D8209781da23fB37b6B59ee15dE7d9841

LINEAR_CURVE: 0x5B6aC51d9B1CeDE0068a1B26533CAce807f883Ee

XYK_CURVE: 0x7942E264e21C5e6CbBA45fe50785a15D3BEb1DA0

PAIR_FACTORY: 0xb16c1342E617A5B6E4b631EB114483FDB289c0A4

PAIR_ROUTER: 0x2b2e8cda09bba9660dca5cb6233787738ad68329

Previous