Whitepaper

A trustless treasury management protocol

Facundo Spagnuolo, Daniel Fernandez, and Bruno Balzani

Abstract

Mimic is a trustless treasury management protocol. It is fully transparent and non-custodial allowing users to allocate third-party assets to different DeFi strategies in a simple, flexible, and secure way.

It provides a set of tools and aggregates many DeFi protocols in one place, making your assets management experience much easier than before. It is meant to be used for DAO treasuries, institutional money, or simply friends & family.

As opposed to other treasury management protocol, Mimic provides an enormous flexibility to embrace this constantly-changing ecosystem making it usable for tons of use cases.

Introduction

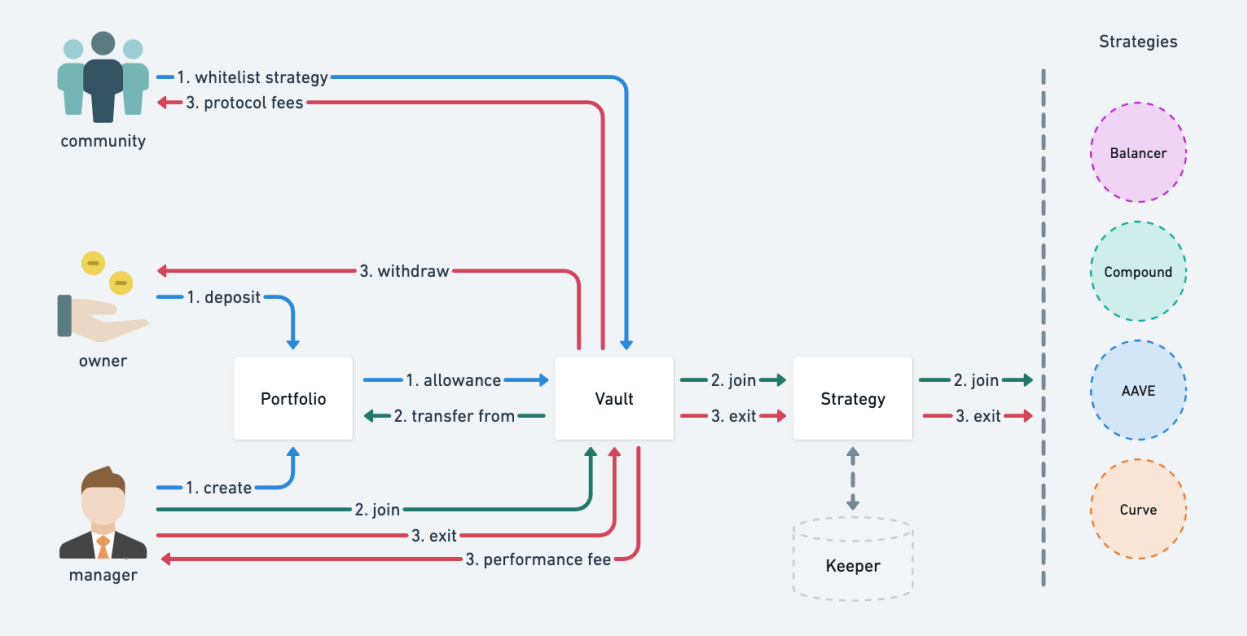
The rise of DAOs as vehicles to drive organizational growth has been exceptional over the last few years. There are many trending DAOs holding +10 billions of dollars in treasury assets. With DAOs gaining popularity, it became a huge question mark how they can **maximize their treasury** while building healthy and sustainable liquidity for their native token.

Additionally, it is well known how hard is to deal with community decisions on-chain. This becomes even worst when talking about investing the DAOs assets. We have seen many DAOs trying to circumvent their own processes to invest their money, at the end trusting individuals to do it carrying huge security risks.

Here is where Mimic comes to play in order to solve **treasury management** with a trustless, fully transparent, and non-custodial solution.

It allows DAOs to allocate their treasury assets to different DeFi strategies while maintaining ownership, and saving time & money. Furthermore, Mimic can also be applied for many other use cases like institutional money or friends & family.

How it works



The are basically 2 main actors here: the **owner** and the **manager**. The **owner** could be a DAO or anyone willing to put their assets to work into different DeFi strategies. This party may not know exactly how to do it, or simply might want to delegate this job to someone else that does it better. This is the case of the **manager**. Managers are the ones who will decide how to allocate these assets based on their best knowledge and interest.

In order for both parties to start working together, the manager creates a **portfolio**, a smart contract with a set of terms that will determine exactly what the manager can do with the owner's assets. In order to accept the portfolio's terms, the owner simply needs to transfer assets to it. It's as simple as that, **there is no further onboarding experience rather than a simple asset transfer**.

At this point, the protocol is fully **trustless**, the manager will only be allowed to allocate these assets to the DeFi strategies that were agreed in the portfolio. And more importantly, it is non-custodial because the owner will always have full control of these funds, the owner is the only one allowed to change the manager or withdraw his assets from the protocol at any time.

> **Managers can now start allocating the owner's assets to different DeFi strategies, without having to ask the owner for permission on every move they want to make. If it's within the boundaries of the terms they agreed on, the manager has full flexibility to operate on the owner's behalf.**

The entire manager's activity occurs on-chain and is fully verifiable, including accounting. Managers can charge different type of fees like deposit, management, or withdraw fees, but also performance fees to guarantee the owner they will only earn money if the owner does as well.

Mimic works with 3 main concepts that will be explained in the following sections: Portfolios, Strategies, and the Vault.

Portfolios

The portfolio holds the list of terms that the manager and the owner have set in order to allocate the owner's assets. There are many concepts that can be customized in a

portfolio:

1. Managers: List of accounts that can manage assets
2. Withdrawers: List of accounts that can withdraw assets
3. Deposit fee: Flat fee the manager can charge to the owner every time new assets are deposited in the portfolio
4. Withdraw fee: Flat fee the manager can charge to the owner every time assets are transferred to the withdrawers
5. Performance fee: Fee the manager can charge to the owner based only on the earnings per strategy
6. Management fee: Fee the manager can charge to the owner based only on the amount of assets that were managed annually
7. Fee collector: Account defined by the manager that will receive the earnings out of performance fees
8. Tokens: Specification of what tokens can be used by the manager. This could be any or a specific list. For example, the manager can be limited on what tokens can be used for trading.
9. Strategies: Specification of what DeFi strategies can be used by the manager. This could be any or a specific list. For example, the manager can be limited on what strategies can be used to allocate the owner's asset.

> **Portfolios can be extended to cover many different use cases like built-in multisig support, KYC integrations, immutable terms, investment clubs, token allocation limits, among others.**

For example, portfolios can support a decision-making process where the manager can propose to change its terms to the owner tied to a timelock. The manager can propose a new strategy of his interest that needs to be approved by the owner before he can start allocating assets.

Another example could be a manager that needs to require the owners to sign a KYC in order to work together and manage their assets. Therefore, the portfolio could have some sort of built-in whitelist. If the assets are owned by a group of shareholders, the

portfolio can implement its own internal accounting to make sure earnings are distributed proportionally between them.

Strategies

These are smart contracts that hold the integration logic to interact with different DeFi protocols. Mimic provides a standard interface to make sure all DeFi protocols can be accessed in a simple way by managers.

Mimic v1 includes a few implementations to integrate different DeFi protocols like Curve, AAVE, Balancer, Compound, Uniswap, and LIDO. However, custom strategies can be developed in a really easy way. Managers have the chance to offer their own DeFi strategies to owners. They just need to follow the standard interface proposed by Mimic.

The protocol also provides a curation process for strategies, this is how managers can make sure they are allocating the owner's assets in a safe place. Initially, the curation process will be done by a governance committee, but it is already planned to become fully decentralized.

> **Mimic opens the doors to a whole world of community developers that want to contribute to strategies development. The protocol and curation process are open to anyone. Managers can allocate assets to any strategy, whether they use the curated list offered by Mimic's community, a custom whitelist, or simply a private strategy developed for their own business.**

Vault

The Vault is the core of the Mimic protocol. It is the nexus between portfolios and strategies holding all the accounting logic. There are five possible actions managers can perform when working with the owner's assets: deposit, swap, join, exit, and withdraw.

Additionally, the Vault provides built-in batching and simulation mechanisms. Batching allows users to enqueue several actions one after the other. For example, a manager

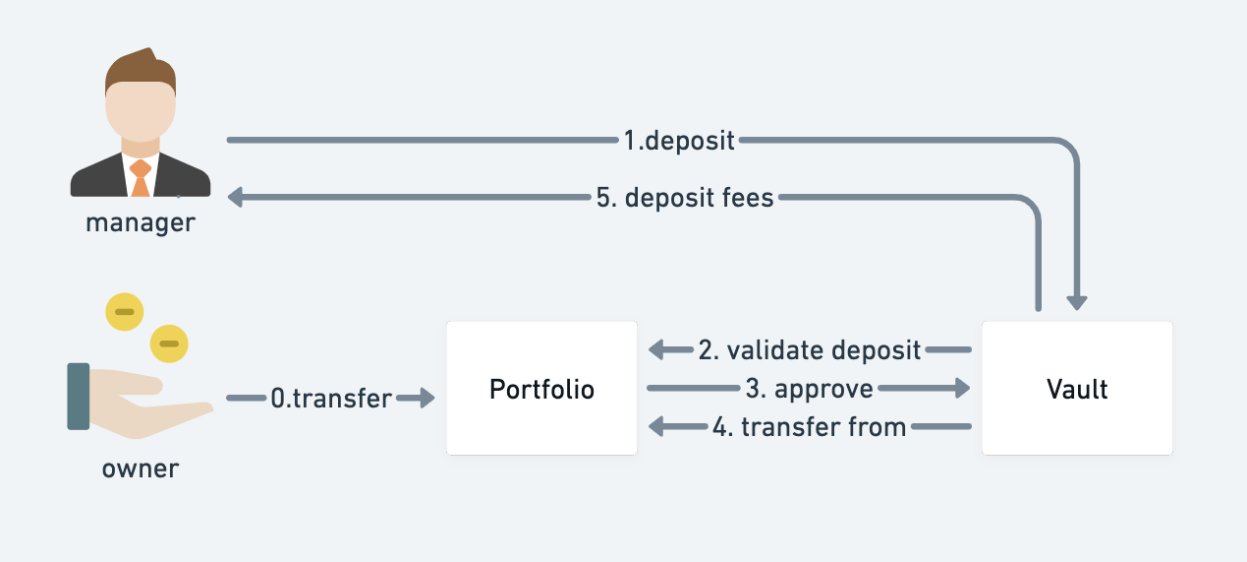
may want to swap some assets and join a strategy with the received amount immediately after. This can be easily done in a single transaction.

Moreover, all these interactions can be simulated on-chain, allowing users to know how much they will get in return. Sometimes it is extremely hard to calculate the outcome of a position if it requires two or three moves beforehand. The simulation process allows managers to know these outcomes easily with no mistakes or rounding errors.

The Vault is where all assets are held before being allocated to the different strategies. Thanks to it, the protocol can make sure that the accounting is properly done. The owner keeps having full control of his funds, no one else can withdraw them from the Vault.

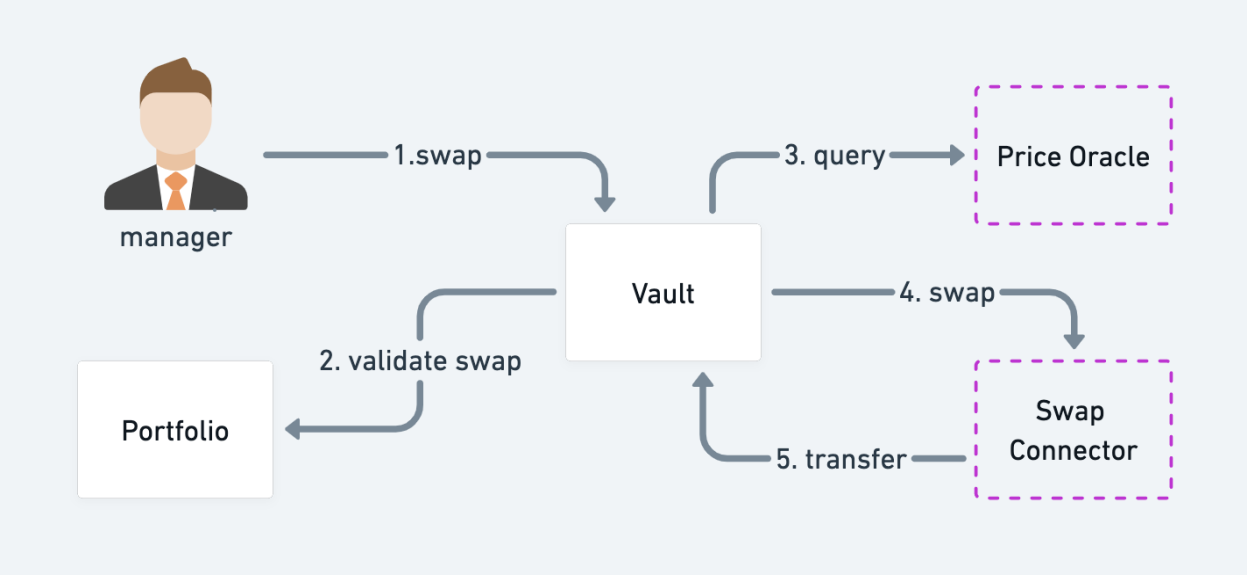
Deposit

This is how managers can bring into play the assets transferred by owners to their portfolios. Here is when the deposit fee is charged and paid out to the fee collector set in the portfolio.



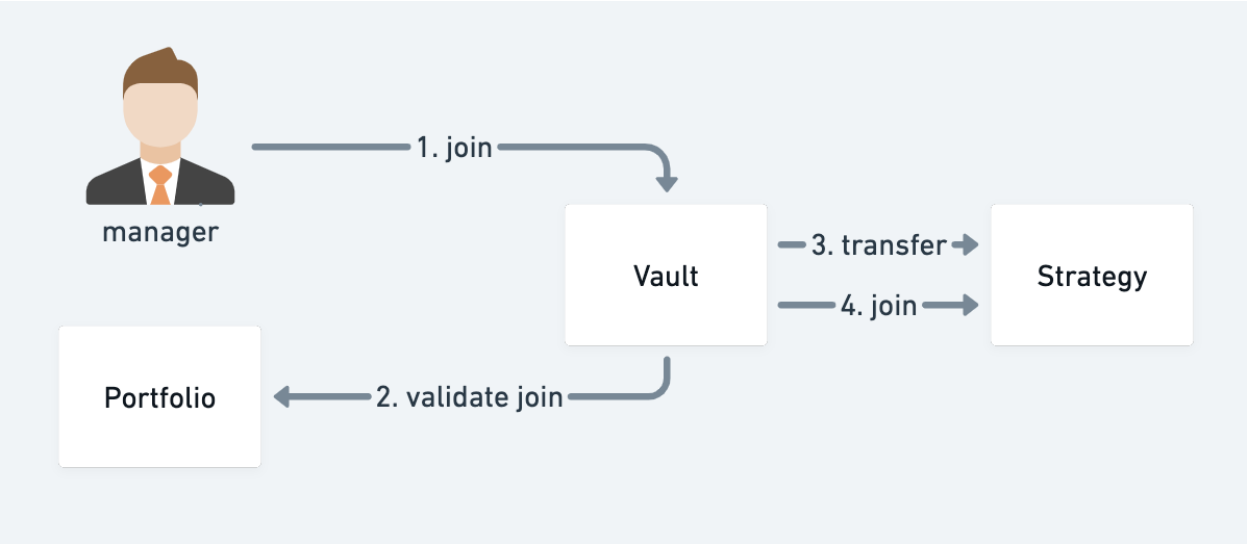
Swap

Managers can swap the owner's assets by other assets of their interest. In this case the Vault will interact with a decentralized price oracle and a decentralized exchange in order to guarantee a safe swap. Before that, the swap is checked against the portfolio, for example the owner could have picked a small list of whitelisted tokens to work with.



Join

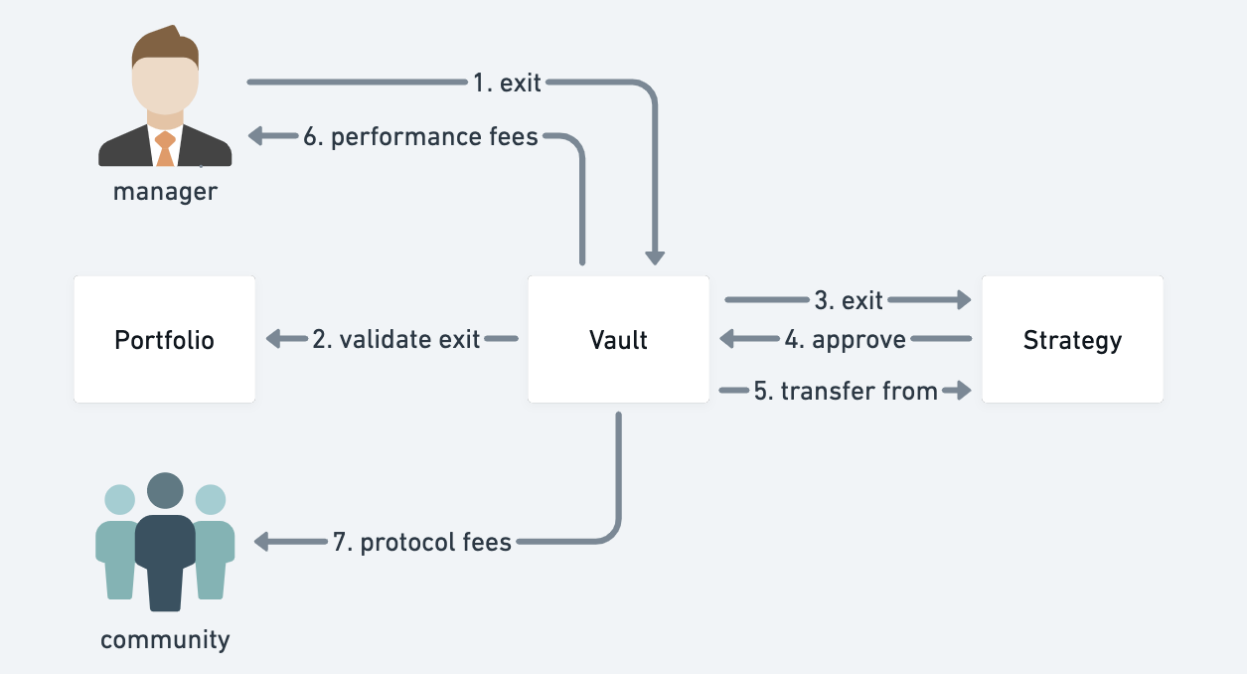
This is how assets are allocated to the different strategies. Always the Vault validates these steps with the portfolio, allowing it to decide whether this action should go through or not. In case it does the Vault ends up transferring the tokens to the strategy and will start tracking its performance to measure gains or losses in the future.



Exit

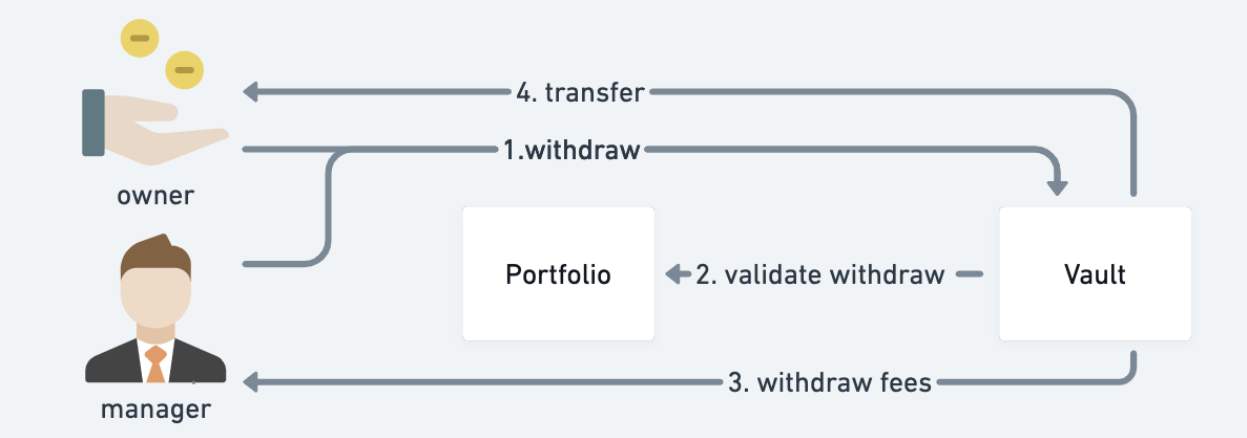
Managers can decide to deallocate assets by exiting strategies. This action will be again checked against the portfolio, any decision making process can be implemented here. Right after that, the Vault will measure gains or losses in order to pay both protocol and

performance fees in case there were some gains. Performance fees will go to the fee collector set in the portfolio, while protocol fees will go to the governance committee that will end up being the Mimic DAO in the future. However, remember that protocol fees are set to zero initially.



Withdraw

Finally, either withdrawers or the manager can decide to withdraw funds. These assets can only go to any of the withdrawers addresses set in the portfolio. Again the manager has the chance to charge withdraw fees to the owner.



> **As you can see, the owner simply needs to interact with the protocol only once. He doesn't even need to call a particular function of the protocol, he simply needs to send assets to the portfolio to denote he agrees with the terms set on it.**

Swap Connector

The Swap Connector is a smart contract that simply assists the Vault by swapping assets. This part of the protocol is held in an external module so it can be replaced by other implementations if desired in the feature. This decision can be made only by the governance committee or the Mimic DAO in the future.

The Swap Connector simply interfaces with external exchanges to swap assets for the Vault. The manager may want to swap the owner's assets in case he prefers holding a specific asset instead of the ones sent by the owner. Additionally, it allows the manager deciding what asset he wants to use to join a strategy.

Price Oracle

The Price Oracle is another external module similar to the Swap Connector that can be used to query the price of an asset in a decentralized way. This module is used along with the Swap Connector in order to have another source of information when validating the prices offered by the Swap Connector. It allows handling slippage in a secure way so the Vault does not have to depend only in what the Swap Connector is saying.

The Price Oracle is also used by some strategies to query token prices. It is useful to guarantee managers can join or exit in a secure way.

This module can be replaced by another implementation if desired as well. This decision can be made only by the governance committee or the Mimic DAO in the future.

Revenue model

Mimic protocol implements what's called protocol fees. The idea of the protocol fees is to charge managers based on their performance fee.

Protocol fees are initially turned off but could be turned on at any time. The only one allowed to do that is the governance committee. It is worth mentioning that protocol fees cannot be grater than 20%, this cap is immutable and set in the smart contracts.

Then, we can define the following revenue model for Mimic:

$$A = gains * protocolFee$$

Where `gains` is the earned yield and `protocolFee` is strictly lower than or equal to `0.2`. Then we can easily calculate the amount to be paid to the manager out of performance fees:

$$B = (gains - A) * performanceFee$$

Where `performanceFee` is set in the portfolio at a strategy level and is strictly lower than or equal to `1`. And finally we can compute the gains the owner will receive after fees:

$$C = gains - A - B$$

Remember this revenue model is only applied when there are gains for a position decided by a manager. If there are no gains no fees are charged at all, neither protocol nor performance fees.

Accounting

Accounting is a hard feature to implement via smart contracts, that is why most managers end up doing bookkeeping off-chain. For a protocol to be completely trustless, accounting must be done on-chain.

Tracking gains is not an easy task. This is because it can represent anything, it can be the yield earned out of a DeFi protocol, liquidity mining returns, the growth ratio between two assets (eg. ETH vs USDC), or a combination of all of them. Summing up, gains can be anything that the owner and the manager agree upon.

This is why, once again, Mimic design decisions regarding accounting were mostly focused on making sure it was flexible enough. The protocol allows the strategy creator to define what do gains mean. To do that, it uses an abstract concept called “value” to

keep track of the strategy gains. By comparing how the value grows, it can determine how gains grow.

$$gainGrowth = currentTotalValue/previousTotalValue$$

Let's see some examples of how yield can be tracked in different DeFi protocols:

AAVE

AAVE is a great example of an easy yield to track . The total value of the strategy equals the amount of the aToken the strategy holds.

$$totalValue = aTokenBalance$$

The amount of aToken that the strategy holds grows over time as it accrues interests from being borrowed on AAVE. It’s growth represents exactly the growth in yield.

Balancer

It is a bit more complex to track yield on Balancer. This is not only because Balancer pools involve many tokens but also because swap fees are collected in any of these tokens. In this case, the total value of the strategy equals the amount of BPT tokens that the strategy holds times the rate of the pool.

$$totalValue = bptBalance * poolRate$$

The pool rate represents the appreciation of one BPT relative to the underlying tokens. Underneath, the pool rate is computed dividing the invariant by the total BPT minted. Because swap fees are the reason why invariant grows within swaps, pool rate can be used as a measure of yield growth in Balancer protocol.

> **In addition, strategies can be scaled to discount any potential expense from gains. For example gas fees, DeFi protocol fees, trade price impact, pools’ impermanent loss, among others.**

Governance

Mimic v1 will include a governance token that anyone could get in order to participate in the Mimic DAO. The DAO will be able to control mainly three things in the protocol: fees, whitelisted strategies, and whitelisted tokens.

Initially Mimic will be governed by a small committee formed by members of the founders, advisors, and investors teams. Once the governance token is deployed, the Mimic DAO will be created and it will be fully transitioned to it, there are no middle steps here, Mimic will be fully decentralized by then.

The governance token will be used for voting to participate in the Mimic DAO. Holders will be able to lock their tokens in order to boost their voting power, usually know as the "voting escrow" model.

Additionally, there will be a liquidity mining program in order to reward investors and managers. The decision of how much goes to the manager can be set in the portfolio, it is not a protocol decision. There will be an initial amount that will be minted to reward early users. Liquidity mining rewards will be paid in the governance token, and how much will be distributed per strategy and portfolio will be decided by the Mimic DAO.

What's next

Integrations

There are tons of extensions that can be built on top of the Mimic protocol. Many different tools can be implemented to make the best use of it. Metrics for owners to keep track of their money, dashboards for managers to measure gains and profit, historic and accounting reports, etc.

Multi-chain

Currently, Mimic v1 has already been deployed to Ethereum and Polygon mainnets. However, we are planning to start working on a more thorough multi-chain support. Ideally, portfolios and strategies could work cross-chain so owners don't have to worry about this. Similarly managers won't need to think about operating on different layers, strategies will become infinitely flexible with trustless bridging.

Contact

https://mimic.fi

hello@mimic.fi

@mimicfi