



## What is Beefy?

Beefy is a Decentralized, Multichain Yield Optimizer that allows its users to earn compound interest on their crypto holdings. Beefy earns you the highest APYs with safety and efficiency in mind.

Through a set of investment strategies secured and enforced by smart contracts, Beefy automatically maximizes the user rewards from various liquidity pools (LPs), automated market making (AMM) projects, and other yield farming opportunities in the DeFi ecosystem.

The main product offered by Beefy Finance are the Vaults in which you stake your crypto tokens. The investment strategy tied to the Vault will automatically increase your deposited token amount by compounding arbitrary yield farm reward tokens back into your initially deposited asset. Despite the name Vault suggests, your funds are never locked in any Vault on Beefy: you can always withdraw at any moment in time.

DeFi applications are unique in the sense that they are permissionless and trustless, meaning that anyone with a supported wallet can interact with them without the need for a trusted middleman. While you have funds staked in a vault, you remain 100% in control of your crypto.

## What is BIFI?

BIFI is the multiutility token of Beefy. The utility is twofold: BIFI stakers earn a part of the revenue generated by Beefy, and both holders and stakers are entitled to vote on important governance decisions.

For all the vaults deployed on every blockchain, Beefy has its multiutility token BIFI at its core. Platform revenue is generated from a small percentage of all the vault profits and distributed back to those who stake BIFI.

The revenue sharing mechanics entail you can stake BIFI to either earn more BIFI in a BIFI Maxi Vault, or earn blue chips like ETH, BNB, FTM, MATIC, AVAX, and more in the BIFI Earnings Pools.

The supply of BIFI is limited at 80,000 tokens and available on various exchanges such as Binance, 1inchexchange and PancakeSwap.

# How to set up a wallet



In DeFi (Decentralized Finance), you are the true owner of your crypto. Your funds sit in a wallet that is controlled and managed by you. Before doing anything, it is important that you always follow these safety practices:

- ✓ • Never share your recovery phrases with anyone, under any circumstances.
- Never input your recovery phrase to a website or app, other than your wallet app when you are importing your wallet on a new device.
- Be wary of fake websites, giveaways, or other malicious acts.
- Download and install only the latest wallet version from official sources.
- Follow the setup guides of your wallet of preference carefully.
- Safely back up your recovery phrases, preferably somewhere offline.

## MetaMask

MetaMask is a very popular browser-based wallet plugin. It supports Ethereum by default, and every other Ethereum compatible blockchain, such as BNB Chain, Avalanche, and Polygon, can be added manually. MetaMask is also compatible with hardware wallets like Ledger and Trezor.

- [Download MetaMask](#)
- [MetaMask Frequently Asked Questions](#)
- [MetaMask Setup Guide by Binance](#)
- [Step-by-step guide by Creatorbread](#)

## Trust Wallet

Trust Wallet is a popular mobile wallet. It is user-friendly as many blockchain networks are preconfigured and it has a built-in DApp Browser. It is however not as configurable as MetaMask.

- [Download Trust Wallet](#)
- [Trust Wallet Community Setup Guide](#)
- [Trust Wallet Setup Guide by Binance](#)

# Funding your wallet



Now you have created your wallet, it's time to fund it with crypto tokens. To make any transaction on a blockchain, such as a trade on a decentralized exchange or a deposit in Beefy's vaults, you need the native gas token to pay for it. For Ethereum, it's ETH, for BNB Chain, it's BNB, etc.

## Fiat On-Ramp



There are several providers that let you buy crypto directly to your wallet by using a bank transfer or credit card. On Beefy, various services allow you to do just that, bundled in one module for an easy overview that lets you pick the best option. You just need to click on the Buy Crypto button on the top of the page to start the Fiat On-Ramp process.

## Centralized Exchange method

Funding your wallet is relatively easy if you already have an account on a centralized exchange, such as Binance or Coinbase. In short, it comes down to withdrawing your crypto from the exchange to your wallet address. In the process, you choose the coin to withdraw and the blockchain network on which you wish to receive the crypto asset.

- [Binance Deposit and Withdrawing guide](#)
- [Coinbase Send and Receive guide](#)

# Connecting your wallet to Beefy

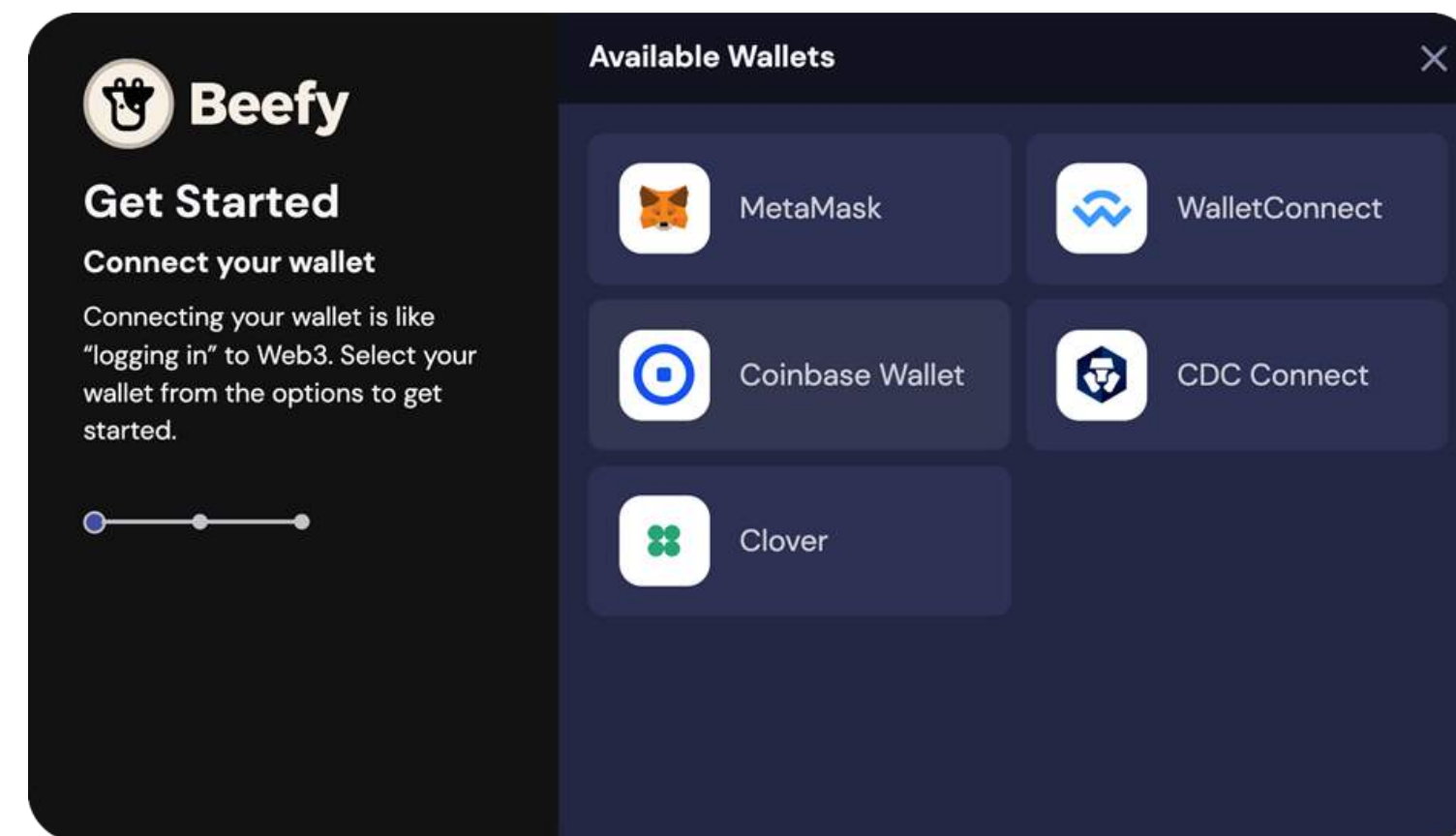
Now that you have created a wallet and funded it with crypto, it's time to connect your wallet to Beefy!

## 1. Click on the "Connect Wallet" button

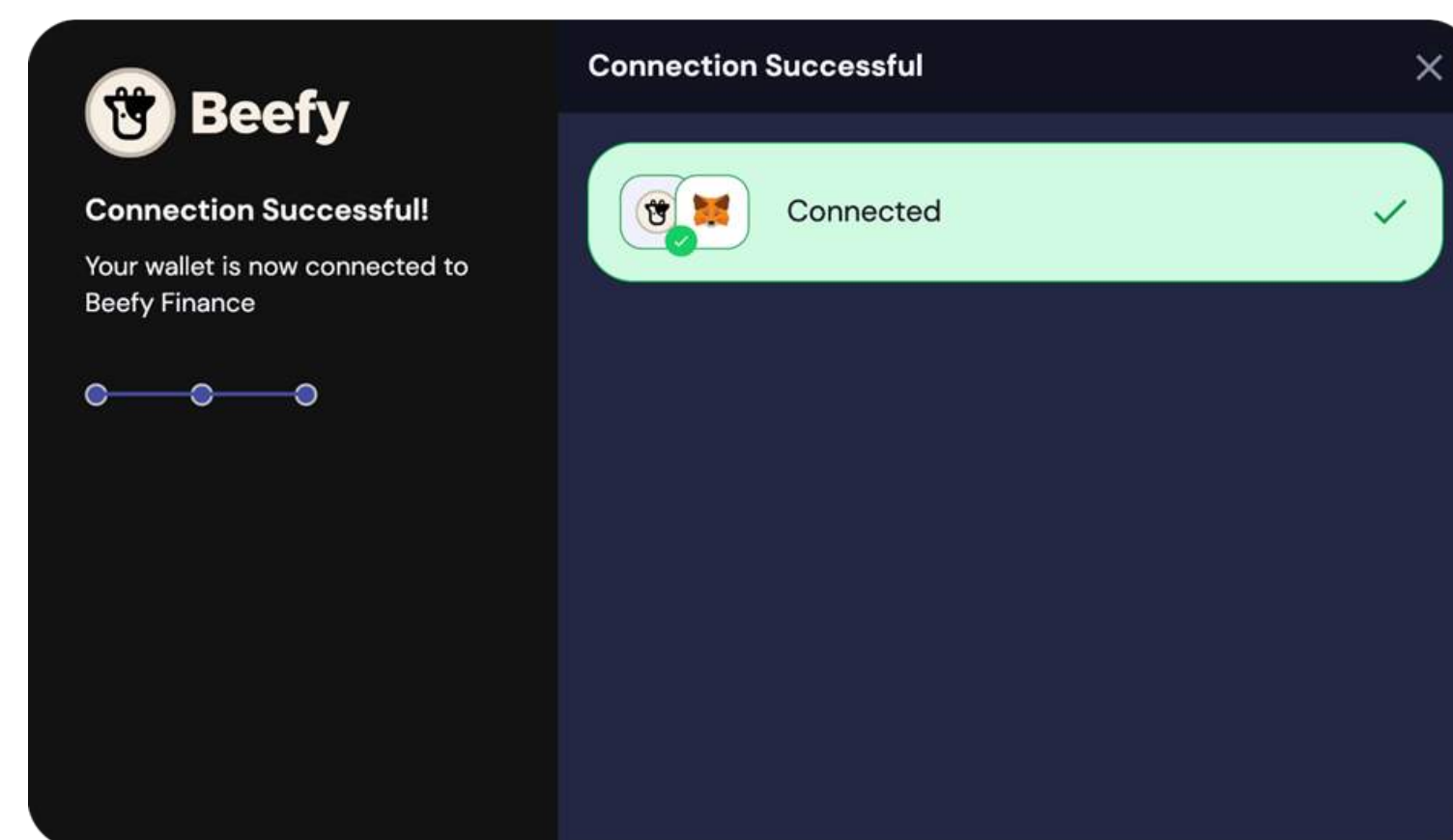
The button can be found at the top of the page:



## 2. Select your wallet from the options



## 3. Approve the connection



All done! 🎉 You are now ready to explore Beefy's vaults and other earnings opportunities.



# Introduction to Beefy

:

Last Update: August 2023

## What is Beefy?

Beefy is a Decentralized Finance (**DeFi**) Yield Optimizer project, that helps users to earn more of the cryptoassets that they love through the magic of autocompounding.

Open-source DeFi applications aim to be permissionless and trustless, meaning that anyone can interact directly without the need for a trusted middlemen. By keeping all code public and verifiable on the blockchain, anyone is able to verify how DeFi works and bypass the need to trust that a service is safe. Beefy leverages these characteristics to deliver hundreds of yield opportunities from across the ecosystem to users in a safe and decentralized manner, through one simple-yet-beautiful interface.

Through its battle-tested [vault](#) and [strategy](#) contracts, the Beefy Protocol maximizes the rewards available to users from liquidity pools (**LPs**), automated market makers (**AMMs**), and other yield farming opportunities. It does this by automatically claiming, swapping and redepositing rewards, unlocking exponential returns through **autocompounding**. By automating the process, Beefy saves time, increases efficiency and circumvents the user risks in manual yield farming, to offer a substantially better experience. In sharing gas fees and aggregating volume, Beefy's contracts are able to compound (or **harvest**) rewards far more frequently than users can, meaning much beefier yields. Overall, Beefy offers a huge advantage to users when compared with farming manually yourself. It's a simple win-win formula.

The Beefy Protocol is governed by its decentralized autonomous organisation (the **Beefy DAO**). The DAO is constituted by hundreds of community members from all across the world, who have a shared appreciation for the magic of autocompounding and a passion for the Beefy project. The DAO is operated using our [\\$BIFI governance token](#), which is used for voting on governance proposals.

## Beefy's History

Beefy was born in September 2020, when a team of 4 founders came together to bring the power of autocompounding technology from DeFi projects on Ethereum over to lower costs chains. The first set of vaults went live on 8 October 2020 on BNB Chain, making Beefy the first yield optimizer on the chain.

In less than a year, Beefy was managing over \$800 million of total value locked, and had reached a \$100 million market capitalisation. Many new contributors and community members arrived, and gradually the founding team stepped back as a new generation of core contributors took over the reigns. By the end of 2022, Beefy was live on 10 different blockchains. A year later, it was 18.

By 2023, Beefy is widely known as one of the OG cross-chain DeFi protocols, with a reputation for indiscriminately building on top of hundreds of protocols. Beefy's contributor team are professionals at scoping out new opportunities, carrying out proper due diligence, and safely and quickly deploying onto the newest and hottest chains and protocols.

## What Makes Beefy Unique?

Beefy differs from other DeFi yield optimizers and aggregators in a few key ways:

1. Beefy's vaults are primarily "single strategy", meaning optimizing just one yield farming opportunity, rather than diluting yield across multiple opportunities;
2. Safety is Beefy's number one priority. All of our products are run through rigorous safety standards through our [SAFU Protocol](#);
3. Beefy prides itself on indiscriminately, quickly and flexibly deploying on top of an enormous range of protocols and blockchains;
4. The Beefy Protocol directly distributes platform revenue back to users who stake \$BIFI in our governance pools;
5. Beefy offers unique strategies that can't be found elsewhere, and is often the first to market with new and exciting yield farming opportunities; and
6. Beefy maintains an enormous network of recognized partners, and has a stellar reputation in the community for its safety and professionalism.

## Autocompounding On Other Chains

Beefy was created with the aim of disseminating the potential of automated yield farming to different blockchains, in order to take advantage of higher transaction speeds and lower gas costs than can be found on Ethereum.

Autocompounding on alternative chains presents a different set of opportunities and challenges: where on Ethereum the timing of events is vital to ensuring profitability, other chains the frequency of low-cost compounding has an enormous impact on the yield. Likewise, low-cost chains encourage farmers to move between different opportunities to maintain high average APY across their portfolio, where moves on chains like Ethereum are expensive and must be carefully considered. This opens the door to more sophisticated and complex strategies on other chains.

Because Beefy's vault offer a fully decentralized and automated solution, they allow users to all sizes and persuasions to access the benefits of DeFi with minimal effort.. We see this as a vital step in levelling the financial playing field, allowing small users to have access to the same opportunities that only the wealthy, professionals and businesses could access elsewhere.

Beefy's goal is to help projects in DeFi grow together, and to spread the potential of decentralized financial technologies across the world. By pursuing this with a complete open-source mentality, a firm commitment to decentralization, and by placing governance openly in the hands of tokenholders, we hope to deliver an unbeatable experience for accessing yield in DeFi.

# Beefy Protocol

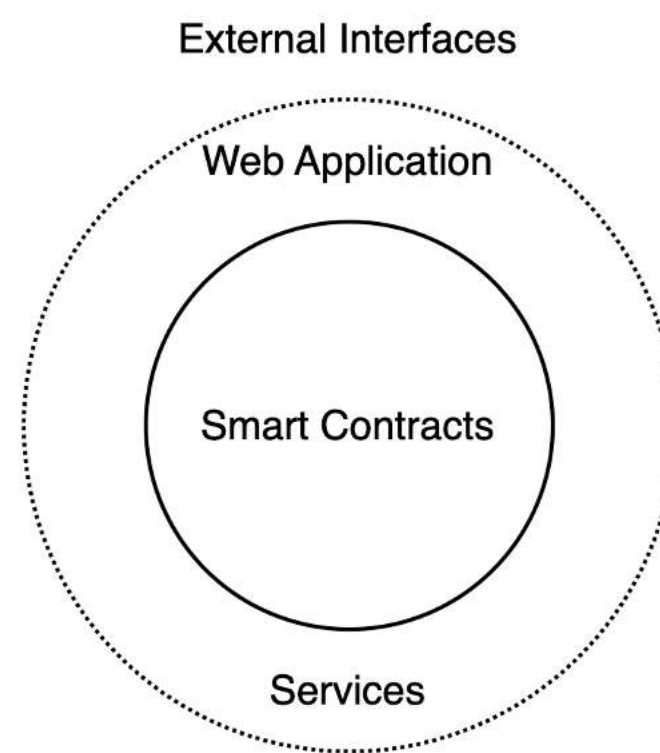
Last Update: August 2023

Beefy is first and foremost an autonomous, decentralized yield optimization protocol. Though the wider project and DAO which surround the protocol are central to Beefy's operations and identity, in fact the protocol functions entirely independently of any such individuals, and will continue to do so on blockchain in perpetuity, even after all other stakeholders are left (even if very, very broken at that stage).

This page provides a short summary of the protocol, designed to onboard novice users as to how the protocol will put their inputs to use. The detail on this page is simplified, and is generally non-technical in nature.

## What is included in the protocol?

As described above, the core of the protocol is the collection of live smart contracts on the blockchain. Beefy's contracts accept user deposits, put funds to work in automated yield farming processes, and then pay out fees to the various on-chain stakeholders in the process. As the smart contracts exist independently of the rest of the project, the protocol can continue operating autonomously, even if all of the supporting stakeholders stopped maintaining them.



A simplified map of the layers of the Beefy protocol.

However, in order for the smart contracts at the heart of the protocol to be maintained and remain accessible to ordinary users, additional services are required. These services typically require ongoing human intervention to deliver and maintain them, and so are typically performed by Beefy's contributor team. These include:

- maintenance of existing smart contracts, such as pausing vulnerable or malfunctioning contracts, or performing live changes or upgrades to facilitate to improve performance;
- development and deployment of new contracts, both to replace and supplement the existing protocol;
- provision of a functional web application or user interface, which assists users in accessing the smart contracts and seeing live information about their functioning (e.g. rates of return);
- maintenance of live servers and databases to store and distribute necessary and relevant data about all other aspects of the protocol, including services like application programming interfaces (APIs); and
- operation of automated contract interactions, such as bots to watch for and trigger profitable compounding events in Beefy's vaults.

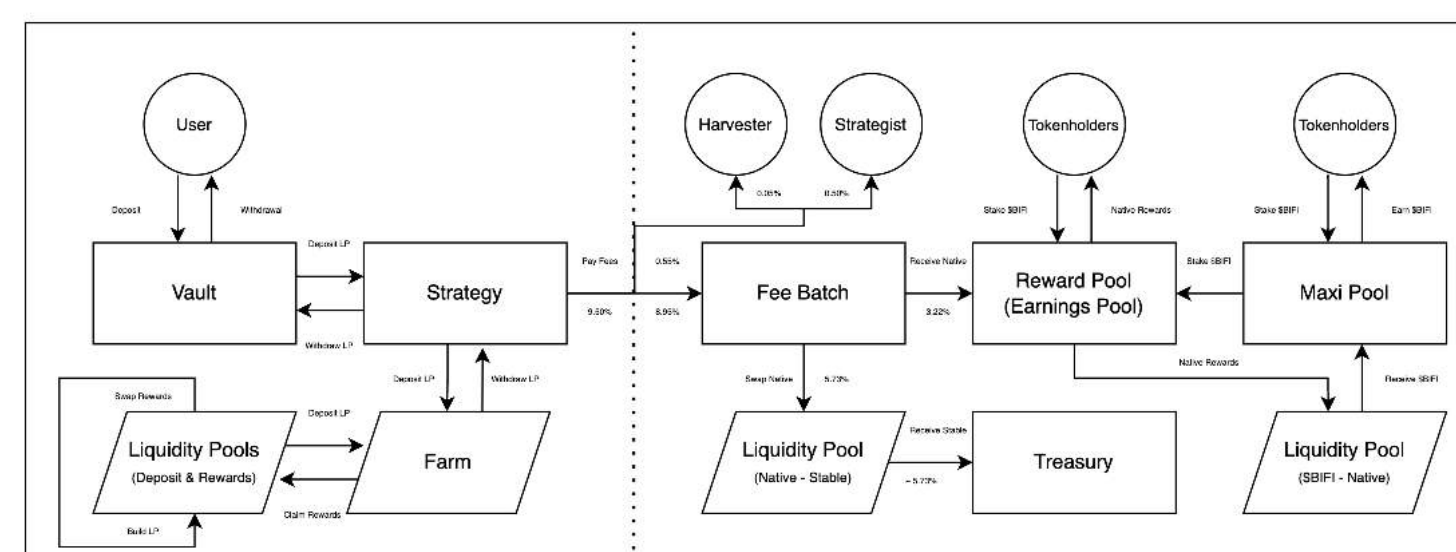
Though the protocol may continue to operate without any one of the above services for a certain period of time, all of these services are required to keep the smart contracts of the Beefy protocol safe, operational, accessible and comprehensible.

Beyond these internal services, external services often build on top of Beefy's protocol to deliver access to its products to users elsewhere. This can include other smart contracts (which deposit funds or act as on-chain stakeholders in the protocol), other web applications (which may point users directly to Beefy's smart contracts) and other information services (e.g. user dashboards, or comparison sites like DefiLlama). By virtue of being external to Beefy, all of these services are not considered to be a part of Beefy's protocol.

## How does the protocol work?

The protocol involves the transmission of funds through various wallets and smart contracts to deliver on its two core functions:

1. To perform automated yield farming, and to optimize the yield received by users from that process, such that they earn more with Beefy than they would elsewhere or doing the same process themselves; and
2. To reward and incentivize the on-chain stakeholders in the protocol for supporting and facilitating its ongoing operations.



A simplified map of the Beefy protocol.

As described in [# What is included in the protocol?](#), the protocol can be thought of simply as just the smart contracts which exist on the blockchain to perform yield optimization. These are reflected in the middle row of the above map, and incorporate our Beefy Vault products (comprised of [Vaults](#) and [Strategies](#)), and then the fee batch contract, reward pool and maxi pool, which together distribute the protocol's fees.

However, in order for each of these contracts to function, the protocol is built on top of other contracts to facilitate the farming process. If all the underlying liquidity pools and farms which Beefy is built on were withdrawn and not replaced, Beefy's contracts may still exist, but no yield or fees would be being generated.

A summary of the core yield farming process is provided in the [Strategies](#) section, so is not repeated here. This process in the left side of the map revolves entirely around the user's deposit and the generation of yield on that deposit through farming and swapping farm rewards.

Each time the yield farming process compounds the rewards, a small fee is charged on the rewards earned, as detailed in [Beefy Fees Breakdown](#). The strategy pays out that fee in the chain's native token directly to the harvest caller (i.e. the wallet which calls the `# harvest()` function), the strategist (i.e. the user who deployed the Beefy Vault), and the fee batch. The role of the fee batch is to accumulate fees from lots of strategies, and then process larger quantities (batches) altogether to efficiently distribute the remainder of the fees.

When the fee batch is sufficiently full, it will also be harvested, which sends a proportion of the fees to the "reward pool" (also known as the Earnings Pool on the web application), and the remainder to the Beefy [Treasury](#), by way of a liquidity pool to swap the native tokens for the Treasury's stablecoin of choice.

The reward pool holds \$BIFI deposits on behalf of tokenholders, and pays out fees as rewards to the tokenholder in proportion with their holding. The default position is that rewards are paid out in the same native token, though the BIFI Maxi Vault was designed and implemented to automatically swap those native rewards into more \$BIFI tokens, in order to facilitate an autocompounding effect for the user.

## What is \$BIFI?

\$BIFI tokens are revenue shares in Beefy, through which holders earn profits generated by Beefy when staked, and are entitled to vote on important platform decisions.

\$BIFI, otherwise known as BIFI, is the native governance token of our project. By staking it in a BIFI Maxi vault, the BIFI earnings pool or any BIFI liquidity vault, or by simply holding this token in your wallet, users can take part in the DAO decision-making governance processes of Beefy. It does not matter on which chain you hold or stake BIFI, since the governance snapshot is multi-chain compatible. Governance proposals are submitted on <https://vote.beefy.finance/#/> and users are encouraged to vote. Users do not need to un-stake their tokens to participate in the voting process. This incentivizes much more voter participation as it means users don't miss rewards. More details of the voting procedure are available on the [Governance page](#).

For all the vaults deployed on every blockchain, Beefy has its native governance token \$BIFI at its core. Platform revenue is generated from a small percentage of all the vault profits and distributed back to those who stake \$BIFI. The revenue sharing mechanics entail you can stake \$BIFI to either earn more \$BIFI in a BIFI Maxi Vault, or earn blue chips like \$ETH, \$BNB, \$FTM, \$MATIC, \$AVAX, and more in the BIFI Earnings Pools.

After an initial distribution period of around two months back in Q4 2020, 72,000 tokens were supplied to the community with 8,000 being locked for the founding team. All 80,000 BIFI are officially in circulation as of July 2022. The distribution via the "governance pools" and detailed info about the timelocks are found [here](#).

\$BIFI staked in the BIFI Maxi vault allows users to accumulate more \$BIFI. In order to distribute BIFI to anyone who has staked in the BIFI Maxi vault, \$BIFI buy-backs from the open market are performed as the token is non-inflationary and will not mint any more BIFI tokens, ever.

To get your hands on BIFI, head to one of the many liquidity pools or exchanges.

## How is the protocol revenue distributed?

All revenue generated on the platform from vault fees is sent to and handled by the RewardPool smart contract in the form of the native token of the chain. Anyone who stakes their \$BIFI in either a BIFI Earnings Pool or a BIFI Maxi vault receives their proportional share.

## What's the BIFI Maxi vault?

The BIFI Maxi vault allows users to stake their \$BIFI much like in the native token earnings pool, but receive instead their rewards in \$BIFI. By staking their \$BIFI, each participant converts and compounds their share of the protocol's revenue into more BIFI tokens. As no more BIFI tokens are to be minted, these are provided to stakers by buying BIFI from the open market with the native token of the blockchain.

This Vault is for users that want to convert and compound their share of the protocol's revenue in more BIFI tokens. As BIFI has no inflation, the strategy market buys BIFI with native token rewards.

## Where can I buy \$BIFI?

\$BIFI can currently be found in several decentralized exchanges including [1inch](#) and [PancakeSwap](#). Buying \$BIFI usually involves exchanging it for another crypto-coin or token through a liquidity pool.

## How do I use my \$BIFI?

To stake your \$BIFI, head over to the [Beefy site](#) and search for our Maxi or Earnings Pool vaults. You can also deposit your \$BIFI into LP positions either by finding an LP vault on the Beefy site (and zapping your \$BIFI into it), or by heading to one of our partner Dexes and investing in the LP directly on their site. Finally, to use your \$BIFI for voting, head over to our [Snapshot site](#) and read through the [Governance page](#) for more details.

# Contract Addresses



## Contract Addresses

The BIFI token is deployed on all chains where vaults are live. To switch to another chain, use [this guide](#) and to bridge BIFI between chains use [this guide](#). Contract addresses below:

- BSC: [0xca3f508b8e4dd382ee878a314789373d80a5190a](#)
- HECO: [0x765277eebeca2e31912c9946eae1021199b39c61](#)
- Avalanche: [0xd6070ae98b8069de6b494332d1a1a81b6179d960](#)
- Polygon: [0xfbdd194376de19a88118e84e279b977f165d01b8](#)
- Fantom: [0xd6070ae98b8069de6b494332d1a1a81b6179d960](#)
- Harmony: [0x6ab6d61428fde76768d7b45d8bfeec19c6ef91a8](#)
- Arbitrum: [0x99c409e5f62e4bd2ac142f17cafb6810b8f0baae](#)
- Celo: [0x639A647fbe20b6c8ac19E48E2de44ea792c62c5C](#)
- Moonriver: [0x173fd7434B8B50dF08e3298f173487ebDB35FD14](#)
- Cronos: [0xe6801928061CDbE32AC5AD0634427E140EFd05F9](#)
- Fuse: [0x2bF9b864cdc97b08B6D79ad4663e71B8aB65c45c](#)
- Metis: [0xe6801928061CDbE32AC5AD0634427E140EFd05F9](#)
- Aurora: [0x218c3c3D49d0E7B37aff0D8bB079de36Ae61A4c0](#)
- Moonbeam: [0x595c8481c48894771CE8FaDE54ac6Bf59093F9E8](#)
- Optimism: [0x4e720dd3ac5cfe1e1fbde4935f386bb1c66f4642](#)
- Oasis (Emerald): [0x65e66a61D0a8F1e686C2D6083ad611a10D84D97A](#)
- Kava: [0xC19281F22A075E0F10351cd5D6Ea9f0AC63d4327](#)
- Ethereum: [0x5870700f1272a1adbb87c3140bd770880a95e55d](#)
- Canto: [0x765277EebeCA2e31912C9946eAe1021199B39C61](#)
- zkSync: [0x44aa3eedd3214ddd02e8b3fe1e8ae4cac452a2e0](#)

On all these blockchains, you can stake \$BIFI in the native staking pools to earn the blockchains respective token or earn more \$BIFI in a BIFI Maxi vault!

## Token Holders

The majority of BIFI tokens are staked in smart contracts, [BscScan](#) has a detailed list of the top staking addresses: [Token Holders](#).



# What is Beefy?



Beefy has been called many things...



- "The compound interest opportunity of a lifetime"
- "The safest and best way to get started in DeFi"
- "Passive income on steroids"
- "Just a really good, super decentralized investment"

***But in practical terms, Beefy is a yield farming optimizer that removes the daily actions and regular fees associated with manual optimization.***

What you're left with is very safe funds and a very good return of investment (ROI).

For those new to decentralized finance (DeFi), yield farming is simply a way to make some interest – as opposed to just "gains" – with your crypto holdings.

Money is one thing, but time is the most important asset of all.

***Beefy makes it simple to benefit from the upside of complex farming strategies.***

There are lots of farms to choose from, so Beefy automates and optimizes different investment strategies in the background while you get on with your day.

The project consists of an anonymous team constantly exploring new methods of optimizing automation to secure the largest yields possible.

As a decentralized project with a crypto-mindset, there is a robust governance system in place to put the decision-making power in the hands of those invested in the project.

**This takes place through governance mechanisms related to Beefy's own fixed-supply, revenue earning token, \$BIFI.**

# The Big Beefy Opportunity



There are two types of people in this world: those who have a vague desire to achieve their financial goals, without ever taking any specific and deliberate action to make it happen, and those who "run the numbers" and are ready to seize the opportunity when they see it.

This might sound like a cringe 80's television commercial, but the truth is the strength of Beefy comes from what we BUILD, not what we say.

*So, what have we built?*

Like any other DeFi Yield Farm, Beefy has created an opportunity for its users to both automate AND maximize the ROI of their holdings.

But what is unique to Beefy is a decentralized working hub for people with a vision to come together and build the future of global finance.

Smart contract devs, UI, UX, strategists, statisticians, designers, and artists — anyone can join and contribute (no matter your nationality, sex, or views).

*By investing in Beefy, you are investing in the idea that a group of highly technical individuals can safely, securely and creatively leapfrog the dinosaurs of traditional finance.*

The two main variables in wealth generation are:

Time (how long) and yield (how much).

We'll take care of the yield.

**All you have to do is decide how soon you would like to get started.**

# What Makes Beefy Different?



Investor and crypto evangelist Naval Ravikant was once asked, "What would you be working on if you were 25 again?"

## Naval's reply:

*"An unstoppable, uncensorable social media platform."*

In other words, he'd be working on a technological platform that puts science and code ahead of cronyism and censorship. This is the foundation of Beefy.

Some other values we align ourselves with:

*Trustless, open-source, decentralized, scalable, transparent, community-driven, cross-chain, autonomous, sustainable, innovation, healthy treasury, contribution rewards, developer bounties, safety.*

In practical terms, our sustainable tokenomics mean the day-to-day is unrelentingly focused on the product. We provide the greatest variety of vaults and the highest number of chains. Users can request vaults directly from our developers on Discord and the time it takes to answer these requests is very low.

At the time of writing, we have more than a dozen smart contract developers (and growing) carefully testing and reviewing the vaults, investment strategies and smart contracts before public release.

## Safety is everything.

And we offer this while saving you time and energy through automation.

***Be first. Be safe. Be Beefy.***

# How Does Beefy Work?

:

We know that entering the crypto and DeFi arenas can be confusing and disorientating...



We know that entering the crypto and DeFi arenas can be confusing and disorientating. We know that many have lost money in altcoins and have high levels of scepticism about this new opportunity.

So let's break down what Beefy does in language as close as possible to how the current financial world works.

## 1. You know what interest is:

You have \$10,000 in your bank account and you earn a 12.5% annual return on your investment. After 12 months, you receive a deposit of \$1,250 as an interest payment. After five years, you have \$16,250.

## 2. You know what compound interest is:

It's interest on interest. You get interest on the initial amount you put in, and you get interest on the interest already accumulated from the previous years.

So with compound interest, after 5 years you won't have \$16,250, you'll actually have \$18,020.

So far, so good.

## *But what if there was a "third way"?*

This is the question we asked ourselves as programmers.

## 3. Here's what Beefy interest is:

When you take your crypto and stake it on Beefy we find ways to add to the compounding of your asset. If one of the platforms we use gives away a promotional coin on top of any interest, then we take that promo coin and sell it for more of what you staked.

***The result is a significantly higher overall annual return.***

When Beefy combines your 12.5% annual compounding interest with the 14.2% interest of another site's promotional coin, you get 28.02% APY on Beefy.

Beefy's BNB Venus vault is doing just that. At the time of writing, you get 28.02% APY for your BNB.

**After five years, you won't have \$18,020, you'll have \$34,386; all in the asset you actually staked.**

# Beefy Fees Breakdown

Anyone familiar with the fee structures typical of traditional finance will tell you fees matter.



## How much do fees matter?

The answer can be hard to wrap your head around.

- \$1M invested for 30 years at 8% with a 1% management fee yields \$7.62 million.
- \$1M invested for 30 years at 8% with a 2% management fee yields \$5.74 million.
- \$1M invested for 30 years at 8% with a 3% management fee yields \$4.32 million.

## Now, let's throw out modesty for a minute.

Unlike some platforms flashing their APYs for your attention, there are no catches on Beefy.Finance. For example, they'll promote an APY, but won't mention there's a penalty fee if you withdraw early.

Or the APY is given as a spectacular headline number, but the small print is that you have to *manually* compound every single day to get that number.

## At Beefy we're proud to be doing things a little more transparently.

*With our vaults, performance fees are included in the APY.*

## So what you see is exactly what you get.

Our vaults on Beefy charge a fixed performance fee structure on their harvest rewards. As described in [# What is the vault fee structure?](#), these fees are distributed back to \$BIFI stakers, the Beefy [Treasury](#), our strategists and the user that harvests the vault. They are the main source of revenue for the platform.

## Here's what a typical vault looks like:

- 3.0% is distributed back to \$BIFI stakers;
- 0.5% is allocated to the Beefy treasury;
- 0.5% is awarded to the vault strategist; and
- 0.05% is awarded to the one calling the harvest function.

Following the passage of [BIP:45], Beefy has introduced a maximum performance fee structure of up to 9.5%. Where this is applied to new vaults, here's what it typically looks like:

- c. 3.22% is distributed back to \$BIFI stakers;
- c. 5.73% is allocated to the Beefy treasury;
- 0.5% is awarded to the vault strategist; and
- 0.05% is awarded to the one calling the harvest function.

For the rare vaults which do not [Harvest on Deposit](#), we assign a withdrawal fee of up to 0.1% to each vault to protect bad actors from abusing the vaults with too much flipping. We share this amongst all the other stakers in the vault.

Finally, for users of our Beefy ZAP V2 tool, we charge a 0.05% zap fee on your deposited amounts when entering or exiting a vault. These fees are returned to the Beefy treasury by way of a intermediate batching treasury, which allows fees to be aggregated and swapped into stables before being deposited. See [How to use Beefy ZAP](#) for more details on the ZAP V2 tool.

Apart from the fees fully listed above, anyone using Beefy should also remember the network transaction fees when adding or removing funds. These small fees go to the operators keeping the blockchain running, not Beefy.

## Bottom line: fees matter.

When a Beefy vault offers you an investment with 14.46% APY, that's always with fees already factored in...

\$1M invested for 30 years at 14.46 % APY yields...

\$57,492,639 million

## With Beefy, what you see is what you get.

# Why Beefy Beats Your Bank

:



A question the wealthiest people in the world obsess over:

**"What is the biggest opportunity for investors right now?"**

It would be imprudent to mention more than superficial details, but it's common knowledge that there are financial benefits, investment vehicles and tax advantages accessible to high net-worth individuals that are closed to everyone else.

*What we can talk about is how traditional banks work, and demonstrate clearly why Beefy.Finance is a better option.*

When you give money to a bank, it lends that money out to other people at a higher rate than you get for "saving".

**The bank pockets the difference as revenue.**

And that money is used to pay for things like staff, rent, call centers, security guards and a myriad of other operating expenses that may or may not be justified.

**Meanwhile Beefy is a decentralized autonomous organisation, collectively owned and managed by its users.**

Decisions are not a top-down event. So instead of the earnings going to a small clique of champagne-quaffing executives, they can only be redistributed with the approval of the organization. DAOs like Beefy are governed by proposals and voting. Code automates the everyday actions banks pay clerks to execute, while cryptography protects your funds.

**So, there are two things happening here:**

First, a DAO is significantly more efficient than a bank. Second, there is a huge demand for better APY than the 0.1% currently on offer by most institutions, which is significantly less than the current inflation rate of 2%.

When you learn that there is a new and safe way to store your money with an APY that is **at least 70x** as good as your bank...

And you appreciate that this APY is available because DAOs are fundamentally better organized to pass on efficiency savings to their users...

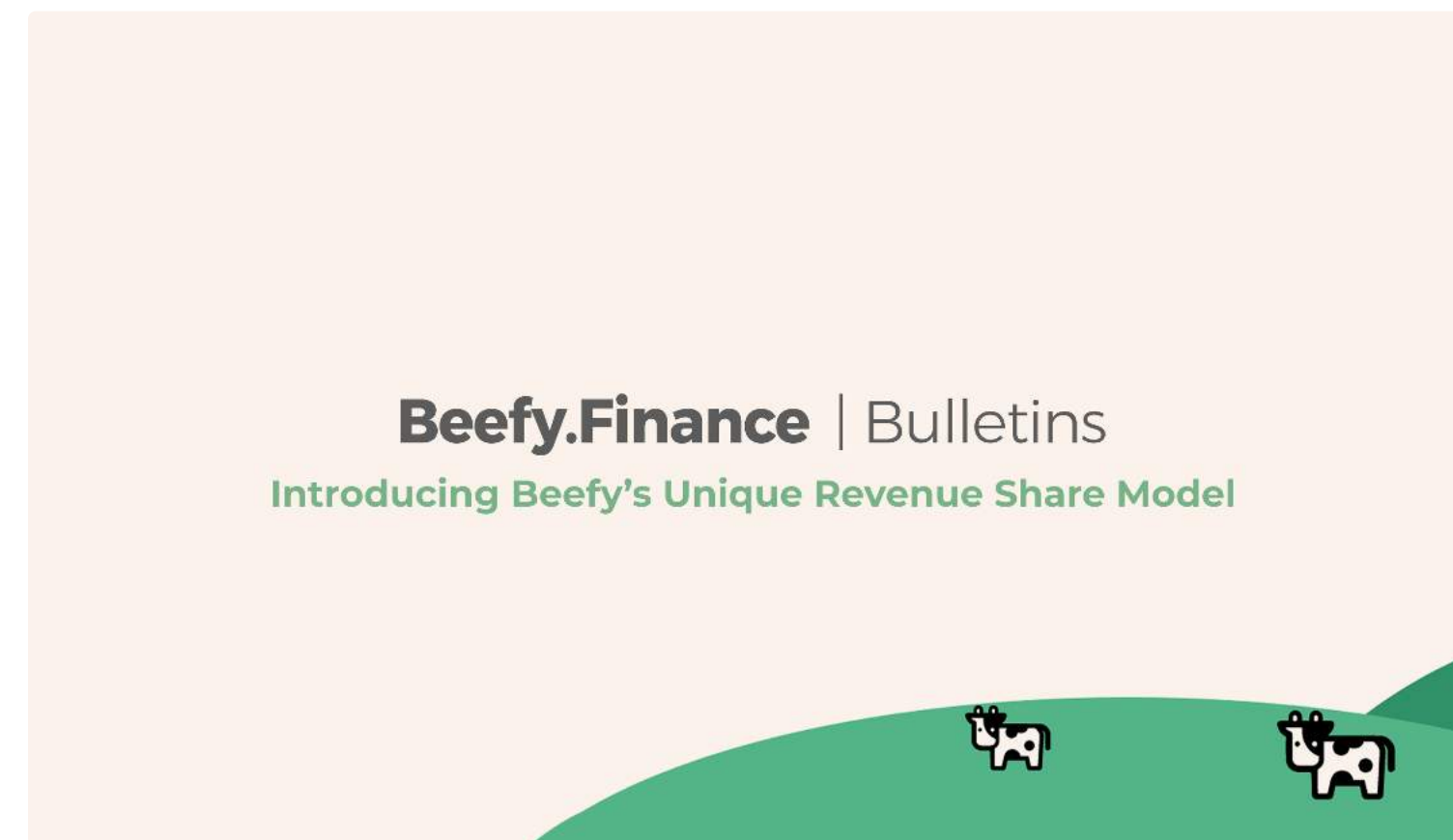
You might reasonably ask how soon you can get started.

And the answer to that is today.

**Because, unlike banks, there's no approval process.**

# Introducing Beefy's Unique Revenue Share Model

By buying \$BIFI and staking in BIFI Maxi, you have something unique



**At long last DeFi adopters are beginning to realize that it's ridiculous to believe so strongly in a better financial system, and then spoil the whole moment with degenerate gambling.**

Hence, the growing popularity of decentralized projects with sustainable tokenomics and platform governance.

They're in a class by themselves.

These projects are **more efficient**, have **better stability and security** and are more likely to attract the **attention of the market**.

Owning \$BIFI not only allows you to participate in platform governance, but also to start earning daily interest payments by staking your holdings in the right place.

Here's where things get *remarkable* — listen closely.

All the APYs you see on Beefy vaults have the platform fees factored in. These fees are used to buy back more \$BIFI from the open market every single day and reward it to those who have their holdings staked in the BIFI Maxi Vault.

So by buying \$BIFI and staking in BIFI Maxi, you have something unique:

*Daily revenue from a non-inflationary revenue token where the "company" buys shares off the market every day, and gives you more of them.*

This creates a financial "flywheel" for your money, where a continuously improving set of repeatable, tactical actions scale with decreasing friction to grow your investment.

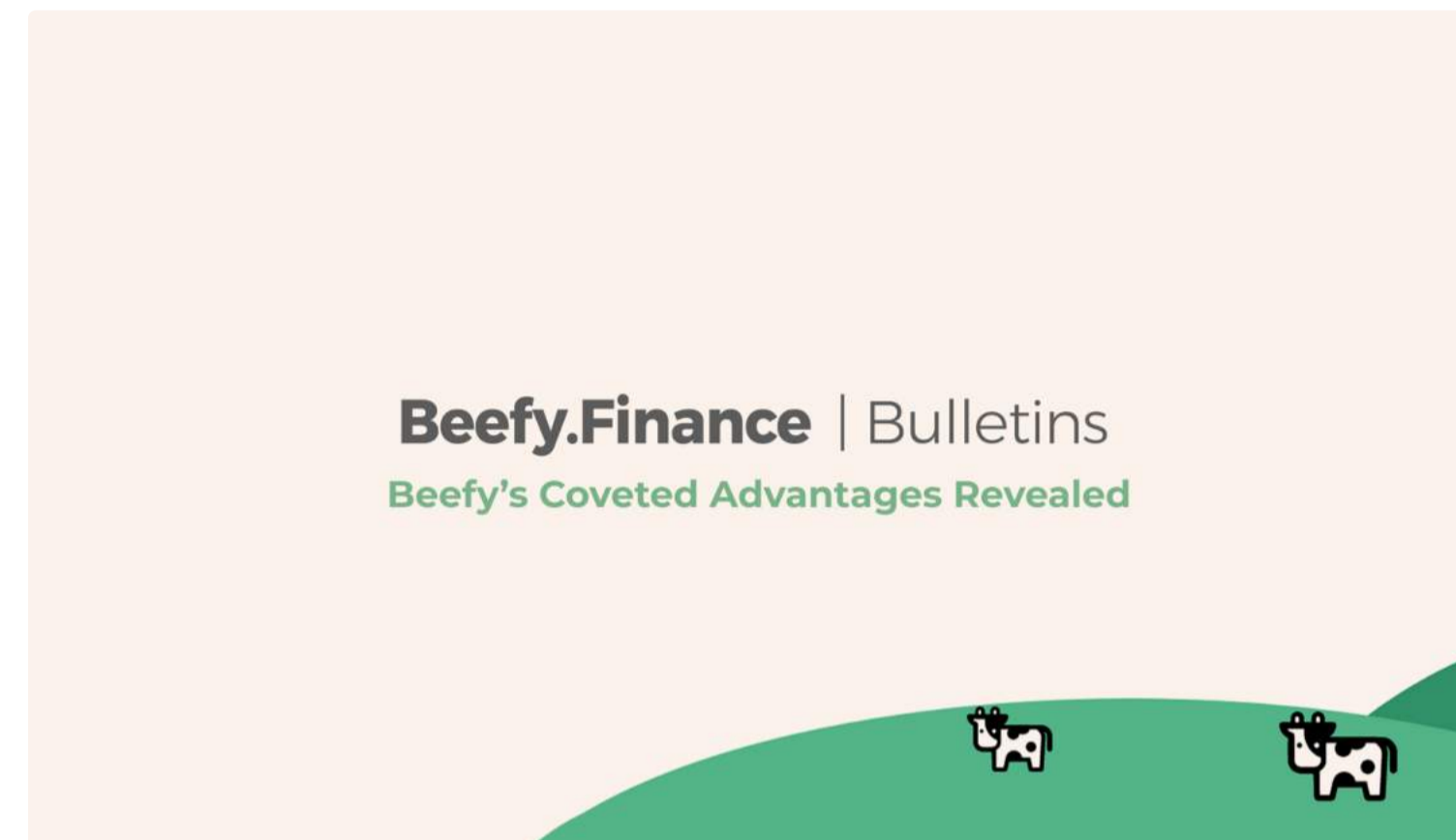
**In other words, yesterday's gains are tomorrow's capital.**

Beefy is being built by a group of dedicated smart contract coders, backed by a community of UI, UX, strategists, statisticians, designers and artists.

That's why the smart money is on [Beefy.Finance](#).

# Beefy's Coveted Advantages Revealed ⋮

How we attract coding and creative talent



*"An organization's ability to learn, and translate that learning into action rapidly, is the ultimate competitive business advantage."*

— Jack Welch, former chairman and CEO, General Electric

**In any activity, there are some advantages that have to do with resources and some that have to do with the absence of them.**

That underdogs often come out on top is because the latter has "unseen advantages" against the former.

At Beefy.Finance, we don't consider ourselves underdogs...

And we generally prefer to let what we build do the talking...

However, a core component of our vision for the future is attracting and incentivizing the best coding and creative talent on the planet to join our community.

Through our unique reward system, our developers receive a percentage of Beefy.Finance's revenue from the vaults they build for Beefy.

Because this exists, we have grown an enormous development team quickly — an army of smart contract developers who are ready to create new vaults almost the same day as we finish our due diligence.

The migration from TradFi to DeFi is a marathon rather than a sprint, but it serves the market to be able to act quickly when we need to.

The marathon means having safety as our number one priority.

The sprint is sustainably spinning up new vaults.

***Code — and the talent that writes it — is the ultimate force multiplier in this arena, and its leverage advantage is compounding at a spectacular rate.***

If you're a smart contract developer who favors future rewards over legacy resources, you know where to find us.



# Vaults

## What is a Vault?

Vaults are investment instruments that employ a specific set of [strategies](#) for yield farming. They make use of automation to continually invest and reinvest deposited funds, which help to achieve high levels of compound interest. By using a Beefy vault to compound your gains, you save thousands of transactions with their associated gas costs, and precious personal time. Instead of manually harvesting and selling rewards, buying more tokens, and reinvesting that continuously, a vault does all that automatically at a high frequency.

Vaults are the core of the Beefy ecosystem. In a Beefy vault, you earn more of the asset you stake in it, regardless if this is an liquidity pool (LP) token or a single asset. For example, vaults where one can stake BTC-BNB LP will result in more BTC-BNB LP over time, effectively growing your share in the liquidity pool and thus allowing for more and more fees and rewards over time.

Despite the name 'Vault' suggests, user funds are never locked in any vault on Beefy. One could always withdraw from a vault at any moment in time. Beefy also does not own user funds staked in vaults. However, it is generally best to view vaults as investment tools to store funds for the medium to long term in order to have the effects of compounding really kick in.

When browsing the vaults on the platform, you will see the annual percentage yield (APY), which takes the frequent compounding into consideration compared to annual percentage rate (APR) which does not. You will also see daily interest percentages and the total amount invested in a vault by all users (TVL). Furthermore, one can see what underlying platform the vault is using as a source of revenue.

Each vault can either refer to a pair of tokens invested in liquidity pools, such as CAKE-BNB LP tokens within the Binance Smart Chain ecosystem, or a single token invested in lending platforms or single stake reward pools. After depositing tokens to a vault, the user is supplied with vault specific mooTokens which represent their share in the vault. We will elaborate on mooTokens in the next section.

Anyone in the Commounity can work together to build new strategies and submit them to the Beefy team for review. However, a new vault will not be accepted if the underlying platform does not adhere to the [Beefy SAFU Practices](#).

Summarizing, vaults can:

- Efficiently execute yield farming strategies.
- Compound rewards into the initially deposited token amount.
- Use any asset as liquidity.
- Provide one asset as collateral for another.
- Manage collateral at a safe level to mitigate liquidation.
- Put any asset to work to generate a yield.
- Reinvest earned profits.

Users can sit back and relax, and watch their investment grow!

## What are mooTokens?

A mooToken is an interest-bearing, tokenized proof of deposit that you will receive at the moment you deposit in a Beefy vault. A mooToken is unique per vault, e.g. you get mooBIFI tokens when depositing BIFI into the BIFI Maxi vault. One can view mooTokens as the receipt of your vault deposit.

**ⓘ** Beefy users should hold on tightly to their mooTokens and not sell or exchange it, since you would lose ownership of your staked vault assets if you did so!

## How do mooTokens earn interest?

Beefy's vaults automatically create more of your deposited asset in the form of compound interest. By holding mooTokens in your wallet, they are increasing in value against its corresponding vault asset. The number of mooTokens in your wallet will remain constant, but the quantity of the vault tokens they can be redeemed for increases. This is also the reason why mooTokens do not 1:1 match with the token amount initially deposited.

## How do I redeem mooTokens for the initially deposited tokens?

Whenever you want to withdraw the tokens that are staked for you in Beefy's vault, you simply initiate a withdrawal transaction to exchange them. The mooTokens are then taken from your wallet and burned, and your deposited assets plus yield will be given back to you.

## What are the advantages of the mooToken system?

Beefy's mooToken system has a few major advantages:

1. mooTokens allow any user to withdraw their fair share of deposited funds;
2. the system allows you to deposit the mooToken receipt to a cold or hardware wallet for ultimate safety;
3. your privacy is maintained, as you remain anonymous to Beefy. Your funds in the vault are not tied to the wallet address from which you made the deposit, since the mooTokens are the only evidence of your share in the vault. Therefore, you could withdraw your share of funds from a different address if you moved your mooTokens to it;
4. mooTokens can have tax benefits. Not only do our mooTokens make bookkeeping super simple, but since you're not selling off your rewards or receiving staking rewards direct to your wallet, (in many jurisdictions) you will not be incurring tax liabilities in the same way you would with farming your own yield; and
5. Lastly, mooTokens can be used as interest-bearing collateral.

## How often do the vaults harvest their profits and reinvest?

Vaults are normally harvested multiple times daily and profits are automatically reinvested (compounded). You can check the harvesting and compounding rate of a vault using [this how-to guide](#).

## Why can't someone just do this themselves?

They could, but vaults help you save on personal time and transaction fees, maintain healthy collateral to debt ratios, self-optimize for the best possible yields, and automatically reinvest earnings. Attempting to do this manually would result in large inefficiencies. At Beefy we like to say: 'Sit back and relax, the vault does all the work for you.'

## What is the vault fee structure?

Most vaults have a performance fee structure, taking a percentage cut of all harvest rewards. This fee on profits is split up and distributed back to BIFI stakers, allocated to Beefy's treasury, sent to the strategist that developed the vault and sent to the one calling the vault's harvest function. These fees are already built into the APY of each vault and daily rate. You do not need to calculate it yourself. The performance fee and the fee structure breakdown are presented inside the Deposit and Withdraw module in a vault.

The performance fee on additional yield, i.e. vault profits, is in part distributed back to BIFI stakers and is the main source of Beefy's platform revenue. A part of it also funds Beefy's treasury which is used to further fund platform development, security and other initiatives. The performance fee was also implemented to promote community engagement and governance participation. A successful and engaged community is critical for our future growth, which in turn rewards platform users even more.

Furthermore, some vaults have a withdrawal fee. The main purpose of this fee is to prevent possible exploits from bad-faith actors. Without the fee, somebody could deposit just before the harvest() function execution and withdraw straight after that event, taking a % of the gains generated by legitimate stakers. Withdrawal fees stay in the vault and are shared amongst vault funds.

Finally, entering or exiting vaults using our Beefy ZAP V2 tool will incur a 0.05% zap fee on your deposited amounts. These fees are returned to the Beefy treasury by way of an intermediate batching treasury, which allows fees to be aggregated and swapped into stables before being deposited. See [How to use Beefy ZAP](#) for more details on the ZAP V2 tool.

## What is harvesting on deposit?

Many of Beefy's vaults "Harvest on Deposit". This means that when you deposit into the vault, you are also calling the harvest function of the vault's strategy. By calling the harvest function, you trigger the collection of pending farm rewards and compounding of those rewards back into the vault tokens for everyone.

Beefy does this so that it is impossible for malicious actors to steal yield, so a withdrawal fee is not required. This greatly benefits long-term investors.

Almost all of the vaults on more inexpensive chains like Fantom and Polygon harvest on deposit. You can tell if a vault harvests on deposit if there is no withdrawal fee.

For depositing, and thus calling the harvest function, you will receive a reward in the form of the native chain token (e.g. WFTM or WMATIC) due to the harvest call fee.

## Harvesting on Ethereum

As transaction fees on Ethereum are expensive, Beefy has introduced a few rules that determine the vault's harvesting frequency.

- Vault TVL is above \$100k: vault will be harvested every 3 days.
- Vault TVL is below \$100k but above \$10k: vault will be harvested every 15 days.
- Vault TVL is below \$10k: community harvest.

Community harvest implies that the harvest function on the strategy contract has to be manually called, and the transaction fees for doing so will not be subsidized by Beefy.

Another rule watches the gas prices on Ethereum. If `maxGasPrice` is 20 Gwei or more, harvests will not be executed as they will become too expensive. This is regardless of a vault's TVL.

The Gelato Off-Chain Resolver that handles the harvests on Ethereum based on the aforementioned rules can be found following this link: [Gelato Automate](#). The smart contract and its parameters, as well as past Executions and Task Logs, are also easily accessible there.

## Harvesting on BNB Chain

BNB Chain also has a harvesting constraint in place:

- Vault TVL is below \$10k and older than 2 weeks: community harvest.

## Does the performance fee get taken out when I withdraw my funds?

No, the performance fees are on profits and are taken every time someone calls the harvest() function.

## Does the vault page show the APY?

Yes. Our displayed APY values reflect the predicted rate earned on a vault in a year. This rate is determined by the underlying platform it uses, the strategy that it is interacting with at the time, the total amount of funds in the vault and also takes into account the effect of compounding. As a unique feature, we have also included all vault fees in the APY calculation. What you see is what you get!

## What risks do the vaults have?

Beefy vaults are audited, but this does not mean that a vault is entirely risk free. Below are some of the general vault risks:

- Assets deposited into the vault have no risk of decreasing in quantity but can decrease in monetary value.
- As with any smart contract, the ultimate risk is that an investor's funds can end up stolen or unable to be withdrawn. The team does take steps to quantify the security risks of smart contracts and will only interact with ones that meet a specific set of requirements after extensive testing to make sure the underlying platform does not contain so called 'rug-pull' functions. For a detailed breakdown of the steps Beefy takes before adding new vaults, please consult [Beefy SAFU Practices](#).

More detailed vault risks, or better yet, information on Beefy's vault safety expressed by the Beefy Safety Score can be found here: [Beefy Safety Score](#).

## Who is in control of the vault?

Each vault and `strategy` is hardcoded, and the code has been built to be immutable, so once they are released, they become unalterable. No one can modify the vaults and strategies.

Modern Beefy vaults do however rely on the standard set out in EIP-1167, known as "minimal proxy" contracts. Minimal proxies reduce deployment costs for repetitive contracts (e.g. vaults) by maintaining the vast majority of core functionality in a single implementation contract. They then configure the individual characteristics of the specific strategy (e.g. the relevant tokens and pools) through the minimal proxy contract - which is a much smaller contract to deploy - which directs instructions through to the implementation contract.

Users should be aware of the distinction between minimal proxy contracts and the proxy pattern used to upgrade contracts. **Beefy's minimal proxy contracts are not upgradeable**, so Beefy cannot take your funds by a sly upgrade. The proxy is only used to reduce deployment costs.

## What are the different vaults?

- **Money Market** : Utilizes lending platforms, such as Venus on BNB Chain or Scream on Fantom, to generate the highest possible yield for these coins (e.g. BUSD, BNB, LINK, DOT, DAI, USDT, ETH, or BTCB).
- **Native Token Farming** : Takes advantage of the high yield on popular farms by depositing another asset to earn, sell and compound profits of the native reward token.

## What will I get out when I make a vault withdrawal?

The default is that you withdraw the token type that you deposited, because at Beefy you earn what you stake. You will get the amount you deposited plus the yield generated (minus a potential vault withdrawal fee). For vaults supporting Beefy ZAP, users can withdraw directly into other assets, including any assets forming part of the in the relevant liquidity pool for ZAP V1, and any of the bluechip, native or stable assets supported for ZAP V2.

## How do LP vaults work?

Liquidity pool (LP) vaults work by reinvesting the fees awarded to LP participants. In return for providing liquidity to the pool, many platforms reward investors with tokens. Our vaults regularly harvest these rewards, sell it, buy more of the LP's underlying assets, and then reinvest to complete the cycle.

This compounds the rewards gained from a liquidity pool. Beefy creates strategies that automate this process, saving you time and gas fees in comparison to farming manually. This is all done for a tiny fee that is distributed back to those who stake in Beefy's governance pool or in the BIFI Maxi vault. A small percentage also goes to the Beefy treasury.

## How often are balances updated in the vaults?

Pending rewards are not reflected in the balance until they are swapped for the initial deposited token. This can vary depending on the strategy running.

## How do vaults get added to Beefy?

New potential vaults can be discussed in our Discord. Our strategists then add the potential investment strategy to our strategy list. A priority is assigned to each new, potential strategy based on its APY, TVL and sustainability. Our developers/strategists then attack the list from top priority to bottom. The official forum is used for submitting actual vault requests.

Then the platform which the vault is potentially going to deposit into, is very thoroughly screened if it has safe smart contracts and no other dangerous traits. For more info on that, please read [Beefy SAFU Practices](#).

## What's your vault naming process?

Each vault on the platform is named after the token that users can deposit in it. For example, the CAKE-BNB LP vault uses CAKE-BNB LP tokens for its investment strategy. A BTC vault uses the BTC token, etc.

Underneath the vault name, you can find the particular vault used for investing the token and farming its yields. For example, Uses: Venus means that that particular vault invests the token in Venus, a DeFi algorithmic money market and synthetic stablecoin protocol.

## How do lending vaults work?

The following applies to: *Aave, Banker Joe, Blitz, Geist, Scream, Venus, and similar lending platforms.*

Most Beefy single asset vaults utilize decentralized marketplaces for lenders and borrowers. By depositing your initial asset in the vault, Beefy deposits it into the lending marketplace and borrows against your token at safe levels of collateral.

The borrowed tokens are redeposited into the platform, and once again used as collateral to borrow more tokens. This cycle is repeated multiple times to generate as much interest as possible from the lending interest and the reward token, which is used to buy more of your originally deposited assets. This strategy is also known as a folding strategy. It is noteworthy that this 'leveraged' multi lending and multi borrowing is only with the deposited vault token, so there is no liquidation risk due to token price swings.

**ⓘ** Transaction fees: because of the multi supply and borrow cycle, the transaction fee for a deposit into or withdrawal from these vaults is generally higher as compared to other vaults.

**⚠** Marketability risk: when the underlying token on the lending platform becomes overborrowed, it can prevent the vault's strategy from deleveraging (unfolding) to accommodate a withdrawal. This usually happens when the market is most volatile, or when there is an ongoing event for which people want to borrow funds from the lending platform. The overborrowed condition will naturally resolve once liquidity returns to the lending platform, a process which can take hours or, sometimes, a few days. Meanwhile, funds always remain safe.

Due to accruing debt/supply interest, one may notice that the deposited token amount may decline ever so slightly in between harvests. After the harvest event, you will see your deposited token amount increase as the yields are compounded back into it. The change in deposited token amount over time of a typical lending style vault looks as follows:



After a harvest event, the yields are added to the deposited token amount

# Strategies

Last Update: August 2023

## What is a strategy?

Beefy strategies are modular smart contracts which direct the user funds deposited into vaults towards liquidity pools and farms in order to generate the yield which Beefy compounds. Where sufficient rewards have amassed in the strategy contract for Beefy to profitably reinvest them, the strategy executes the compounding workflow (or "harvest"), which automatically claims the rewards, swaps them for the principal assets and redeposits them into the liquidity pool and farm.

Strategies are the core product which of Beefy's protocol, as - unlike the vault contract - each strategy is generally unique. Because each strategy involves a different combination of assets, pools, protocols and chains, each requires [individual testing](#) to ensure it is working as intended before it can be pushed into production.

As strategies are the component of a Beefy Vault which interact with external protocols, they are also the component which contains exposure to extrinsic risks beyond Beefy's control (e.g. flaws of the contracts of other protocols). Because of this, strategies contain additional functionality to respond to extrinsic risk, including the ability to "panic" the vault - i.e. withdraw all funds from third party contracts to be held safely in the strategy - or "paused" - i.e. halt the operation of all of the contract's functions.

## How do liquidity pool strategies work?

Beefy's most common product is a standard liquidity pool vault, often built on the Uniswap V2 standard (i.e. a 50/50 balance of two assets with uniform concentration across the pool).

Taking this as an example, the workflow of the strategy starts with user deposits, which can be either the LP token, the underlying assets of the vault, or (with ZAP V2) any supported bluechip asset or stablecoin. The deposit is secured in the vault to handle the deposit/withdrawal workflows, before being transmitted to the strategy.

The strategy then deposits the assets into the liquidity pool, and in turn the LP tokens into a farm, which begins to accrue both trading fees and farm rewards for the user.

The vault then regularly claims the accrued rewards, routes them to a core liquidity pool for swap them back into the underlying asset, and then redeposits the underlying asset into the Beefy vault and strategy.

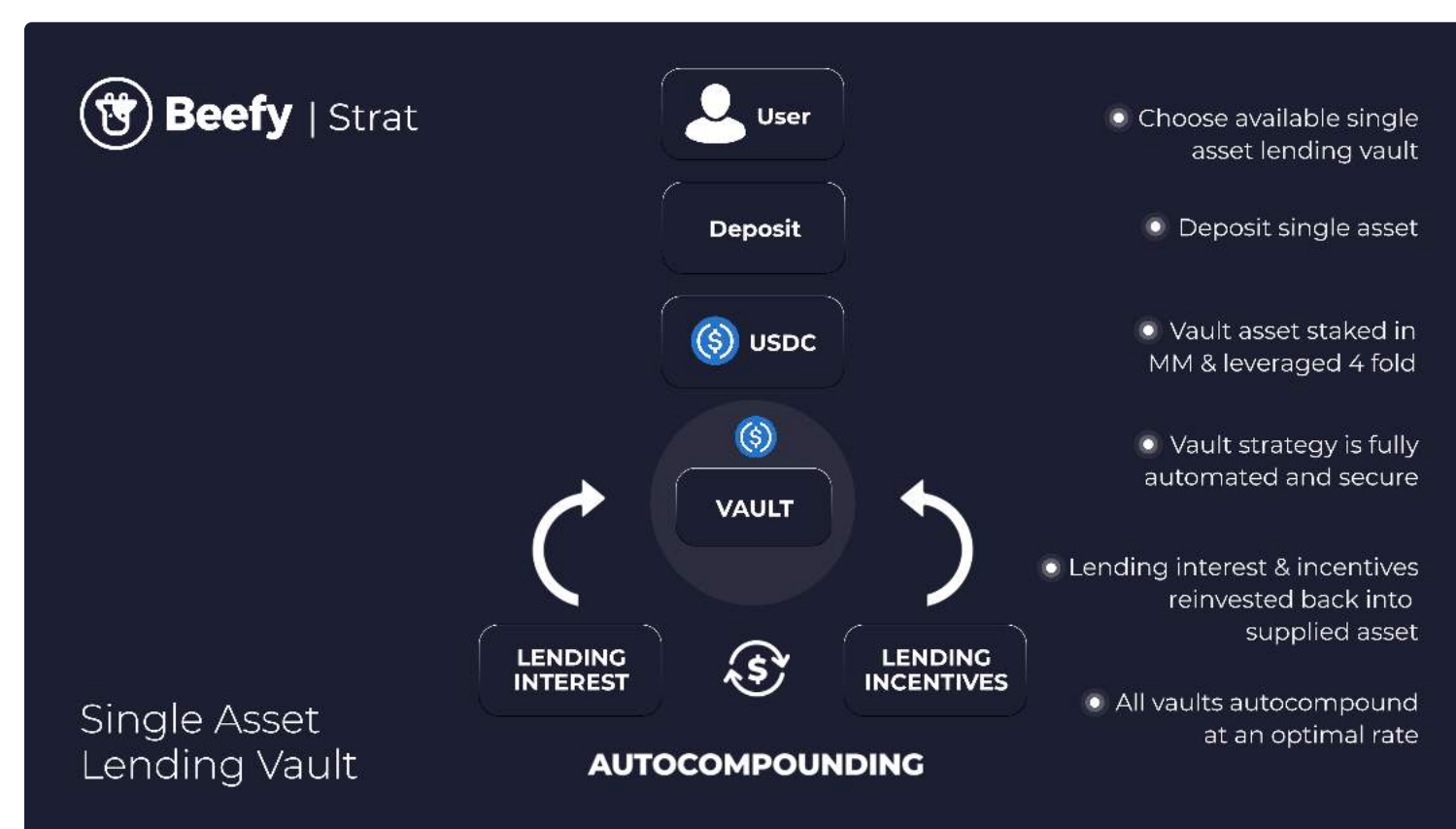


The standard workflow for a typical liquidity pool strategy

## How do lending strategies work?

Standard lending vaults work in a very similar way, taking assets deposited in the vault and deploying them into the lending pool of choice.

Instead of trading fees and farm rewards, lending pools pay out interest on deposits and lending incentives, which need to be redeemed, exchanged and redeposited by the strategy.



The standard workflow for a typical lending strategy

## Who is in control of the strategies?

Each vault and strategy link is hardcoded, and the code has been built to be immutable, so once they are released, they become unalterable. No one can modify the vaults and strategies.

Modern Beefy strategies do however rely on the standard set out in EIP-1167, known as "minimal proxy" contracts. Minimal proxies reduce deployment costs for repetitive contracts (e.g. strategies) by maintaining the vast majority of core functionality in a single implementation contract. They then configure the individual characteristics of the specific strategy (e.g. the relevant tokens and pools) through the minimal proxy contract - which is a much smaller contract to deploy - which directs instructions through to the implementation contract.

Users should be aware of the distinction between minimal proxy contracts and the proxy pattern used to upgrade contracts. Beefy's minimal proxy contracts are not upgradeable, so Beefy cannot take your funds by a sly upgrade. The proxy is only used to reduce deployment costs.

With that said, all vault contracts do contain the functionality to change the strategy where - for example - an underlying DEX replaces its liquidity pool with an upgraded version. Strategy changes allow user funds to be retained in the vault (avoiding imposing migration costs on the user), but allow the strategy to be kept up to date. Strategies are replaced using the `upgradeStrat()` function, and the `proposeStrat()` function as a precursor, both of which emit trackable events to monitor strategy replacements.

## How can I make a strategy?

For users looking to get more involved and become a part of Beefy's team of strategists, you can post and discuss your strategy ideas in Beefy's Discord in the #strategy-devs channel. Please make sure to provide detail of what pool and farm you're considering, what type of protocol they're on and what the APY is. There will be a template to help you get started.

## What is APR and APY?

APR reflects the simple interest rate over a year's time, while APY describes the rate with the effect of compounding.

## Is APY/365 the right way to determine daily gains?

No, the effect of compounded interest is exponential, not linear. A daily compounded interest of 1% would yield 3678.34% a year. The correct formula for daily yield is:  $Daily Yield = ((1 + Annual Yield)^{(1/365.25)}) - 1$ .

## How does Beefy optimize APY?

Beefy automates the entire compounding process, making it close to optimal as possible. The key factor is the frequency of compounding events, which depends on different variables in the system, like current gas prices, rewards accrued and liquidity for swaps. Beefy's sophisticated harvesting automation technology is watching all of the protocol's vaults around the clock, waiting eagerly for the next optimal opportunity to harvest.

# Boost

Earn extra yield on top of your vault earnings!

As The Multichain Yield Optimizer, Beefy always wants to make sure that the users have the best experience possible using Beefy and get the highest APY with the least amount of manual effort. In order to do just that, we have expanded on the initial vault offering with a service called Beefy Boost in which we promote exciting projects on various chains. Together with the promotion, we boost certain vaults with the partner's token in order to give you the best APY.

## What is a Boost?

When you deposit in a Beefy vault, you receive a 'receipt' token prefixed with 'moo' in your wallet. When a Boost is available, you may stake that token in the Boost to receive the extra earnings benefit. You earn extra yield on top of your vault earnings!

## How do I use Beefy Boost?

First, look for a boosted vault in our main app and deposit the tokens that are asked for in the vault. Then, proceed to the Boost section to stake your [mooToken](#) "deposit receipts". Stake these [mooTokens](#) and you are all done! You can easily come back here to check on your earned partner tokens and withdraw at any time.

## How do I see my earned tokens?

Enter the vault where you deposited your [mooTokens](#) and it will show you a nice summary of your earned tokens.

## What are mooTokens?

A mooToken is an interest-bearing, tokenized proof of deposit that you will receive at the moment you deposit in a Beefy vault. One can view mooTokens as the receipt of your vault deposit. Read more about mooTokens under the [Vaults](#) section:

- [What are mooTokens?](#)
- [How do mooTokens earn interest?](#)
- [How do I redeem mooTokens for the initially deposited tokens?](#)
- [What are the advantages of the mooToken system?](#)

## How long will the Boosted vault last?

There is a timer shown in the Boost section for each boosted vault. This is nothing you really need to keep track of since you can always come back after a vault is finished and withdraw then.

## Do I have to manually unstake from the Boost when it is finished?

Yes, this is a mandatory user action when the Boost ends. For safety reasons, Beefy can not move your tokens back to your wallet. You can just come back after the Boost is finished to unstake your [mooTokens](#) together with the partner tokens, at any time, by hitting "Claim & Unstake".

## Can I enter multiple boosted vaults at once?

Absolutely! Just deposit the required tokens in one or multiple of our boosted vaults and then deposit your [mooTokens](#) in the Boost section. Repeat this step for every boosted vault you want to be a part of.

## Are boosted vaults safe to use?

Yes! Beefy always develops with *safety* in mind and has made sure the boosted vaults are completely safe. They are hosted by Beefy, and Beefy has gotten the boost tokens from our partners and uses our own vaults for the reward. The [mooTokens](#) you stake to receive the boost tokens do not leave Beefy.

## If I enter a partner vault with my mooTokens, will I still earn the ordinary vault reward?

Yes! The ordinary tokens you deposited in our main vaults will earn the ordinary rewards and be compounded as usual, even if it is boosted. This is because the [mooTokens](#) you deposit to enter the Launchpool boost are *interest-bearing*. By using the boosted vault, you earn both the vault tokens and partner tokens as a boost.

## Is there any fee on the Beefy Boosts?

Beefy takes a performance fee of 5% from the reward tokens. The fee is deducted when the boost starts and just like the [Vault fees](#), the APR that you see displayed is final so you don't have to account for the deduction.

## What does "Pre-Stake" mean?

This means there is an upcoming boost for this vault. You can stake your mooTokens ahead of time so that you will earn the Boost Rewards as soon the Boost begins.

## How come the APY shown at launch is not the same as it is now?

APY is the "annual percentage yield" and is calculated by compounding your yield interest daily. The daily yield in turn is based on factors such as the reward rate and the total amount of deposited tokens that share the rewards. When more people and in turn tokens enter the pool, the fixed yield is shared by more people (tokens) hence the daily yield will become lower and in turn, lower the APY. In the same way, if people (tokens) exit the vault, there are fewer people (tokens) sharing the fixed reward and the daily yield will increase and in turn, APY will increase.

## Are the promoted project and its tokens safe?

When partnering with a certain project, Beefy always makes an overall due diligence check of the project to get a sense of its sincerity and safety. For this, Beefy follows a stringent set of safety rules, as outlined in [Beefy SAFU Practices](#). Despite the project being safe code-wise, we can never guarantee if the partner token is worth holding. Therefore it is always up to you to make your financial decisions, to make sure that the partnering project is a project that you want to support or not, and that the partner token is one you want to hold or sell. Beefy cannot, and will not take any responsibility for your personal actions.

# Beefy-escrowed Tokens

Last Update: October 2022

## What is Escrow?

An escrow mechanism is where one party holds another's rights or assets in custody for them, pending fulfilment of a specific condition (e.g. safe receipt of goods purchased with those assets). In DeFi, escrow mechanisms involve you providing your tokens to a third party (e.g. Beefy) to allow them to implement some functionality with those tokens on your behalf. As such, all escrow mechanisms are typically built around some token economic ("**tokenomic**") design in the tokens you hold.

For example, a blockchain's native token (e.g. Fantom's FTM) may be used for staking purposes to secure a blockchain by fairly and safely validating its transactions. The tokenomic design allows users to stake their tokens in order to earn transaction fees from users of the blockchain, giving the native token an ability to earn or generate value for the holder. However, staking is complicated and has trade offs, like locking your tokens and limiting your ability to trade in them. So third party escrow solutions (e.g. Beefy's beFTM) have been created to allow you to access the benefits of tokenomics (i.e. staking your FTM) without having to manage the staking yourself, and whilst allowing you to still trade your interest in the locked tokens.

## What is Vote Escrow?

A vote escrow mechanism is a form of tokenomic escrow design which aims to reward long term holders of a protocol's governance tokens with more voting power, to favour their interests in governance matters over short term holders. This is achieved by holders staking and typically locking their governance tokens with the protocol (typically for a return from protocol earnings), thereby limiting their ability to deal in those tokens. In doing so, the holder either unlocks voting rights (which are otherwise not available to holders) or boosts their existing voting power, typically in line with the amount of time that their tokens are staked/locked for. This voting power is often used to direct the economics of the relevant protocol, like by having users vote on how to distribute newly issued tokens as additional incentives among the protocol's liquidity pools.

The vote escrow system was pioneered in DeFi by Curve Finance's CRV and veCRV token design (hosted on Ethereum). Other notable examples of escrow designs include Convex's veCVX (Ethereum), Balancer's veBAL (Ethereum), Frax's veFXS (Ethereum), Mai Finance's eQi (Polygon), Trader Joe's veJOE (Avalanche), Platypus's vePTP (Avalanche), Velodrome's veVELO (Optimism).

## What are Vote-escrowed Tokens?

In most vote escrow systems, the voting power of users is not linearly related to the number of governance tokens held, but instead depends on factors like the length of time which the tokens are locked for. In light of this, the concept of vote-escrowed tokens ("**veTokens**") was developed, to reflect the amount of voting power that a user has arising from their staked/locked tokens. As the factors that impact on voting power change over time, the user's amount of veTokens will also change, even as the number of governance tokens held stays the same.

One point of confusion is that - despite the name - veTokens aren't always represented by an ERC20 token, so aren't necessarily received and held by users in their wallet. This is because veToken economics ("**veTokenomics**") are designed to allow for voting power to change constantly as the input factors (e.g. length of the lock period) change. If veTokens were to be issued to users, they would require constant rebasing (adjusting the total supply and nominal holdings of all users) to maintain a fair record, which would add a lot of additional complexity and cost. Furthermore, as veTokenomics were designed to incentivise long-term holding of governance tokens, implementing voting power in a transferable ERC20 format may undermine that goal, as it would allow long term holders to trade their interests in the same way that short term holders do.

> [Example of veTokenomics](#)

## What are Beefy-escrowed Tokens?

Beefy-escrowed token ("**beTokens**") are wrapper tokens, designed and implemented by Beefy to unlock the benefits of escrow tokenomics whilst enabling our users to trade their interests in the partners' governance tokens. Effectively, we put in place an interim smart contract which can hold the governance tokens, lock them with the partner protocol for the maximum possible lock and gain access to the benefits of the escrow model. In addition, we add extra value by combining these features with issuing a new ERC20 beToken to you which you can then exchange to exit the lock at any time.

Each beToken is uniquely designed around the different veTokenomic model of our partner protocols, so they come with a range of different possible features. These can include: withdrawal reserves; supported DEX liquidity; pegged or free-floating pricing; a Beefy voting process for allocating votes on the underlying protocol; user-led or Beefy-led bribes; and boosts to associated Beefy vaults.

Beefy always support our beTokens with vaults on the Beefy platform, where you can stake your beTokens to earn a return. This may be in the form of an autocompounding beToken Vault (i.e. earn more beTokens) or an Earnings Pool (i.e. earning more of the underlying governance token). In either case, you're free to withdraw from your staking at any time, to trade in your beToken and exit your position.

## What Beefy-escrowed Tokens are there?

In this section, we provide a page covering each of the different beTokens that we currently operate. At the time of writing, these include:

 [beFTM](#)

 [binSPIRIT](#)

 [beJOE](#)

 [beQi](#)

 [beVELO](#)

## Where can I find out more?

This section provides full details of the designs of each of our existing beTokens. You can also raise any specific questions about our beTokens by reaching out to us on the [Beefy Discord](#) server.

# beFTM

Beefy-escrowed Fantom

Note that following the passage of [BIP-67], the current version of beFTM will be gradually deprecated with all existing FTM deposited to be unlocked by 3 April 2024. Beefy's developer team is exploring a replacement V2 beFTM, to be released closer to the end of the deprecation period.

## What is beFTM?

beFTM is short for Beefy-escrowed Fantom. beFTM gives stakers access to maximized Validator Node rewards that typically aren't available to the individual investor without locking FTM for 1 year. The beFTM token is 1:1 backed by FTM and can be staked on the Beefy platform and in farms on major DEXes.

## How does one get beFTM?

To get your hands on beFTM, users must first deposit their FTM in the Beefy Delegator Vault. The FTM is then used to buy beFTM from a liquidity pool or minted 1:1 depending on which is most profitable for the user.



beFTM can also be manually purchased on many DEXes such as Solidly, BeethovenX, SpiritSwap, and SpookySwap. One might need the beFTM contract address for trading:  
`0x7381eD41F6dE418DdE5e84B55590422a57917886`

## How does beFTM delegation work?

The collective pool of FTM is delegated to Beefy's Validator Node and perpetually locked as long as possible (51 weeks) to earn maximum validator rewards. We supply 1/15 of the deposit to our validator so we never hit the delegation limit. By staking our users' FTM together, the Cowmoonity enjoys a much higher rate of return.

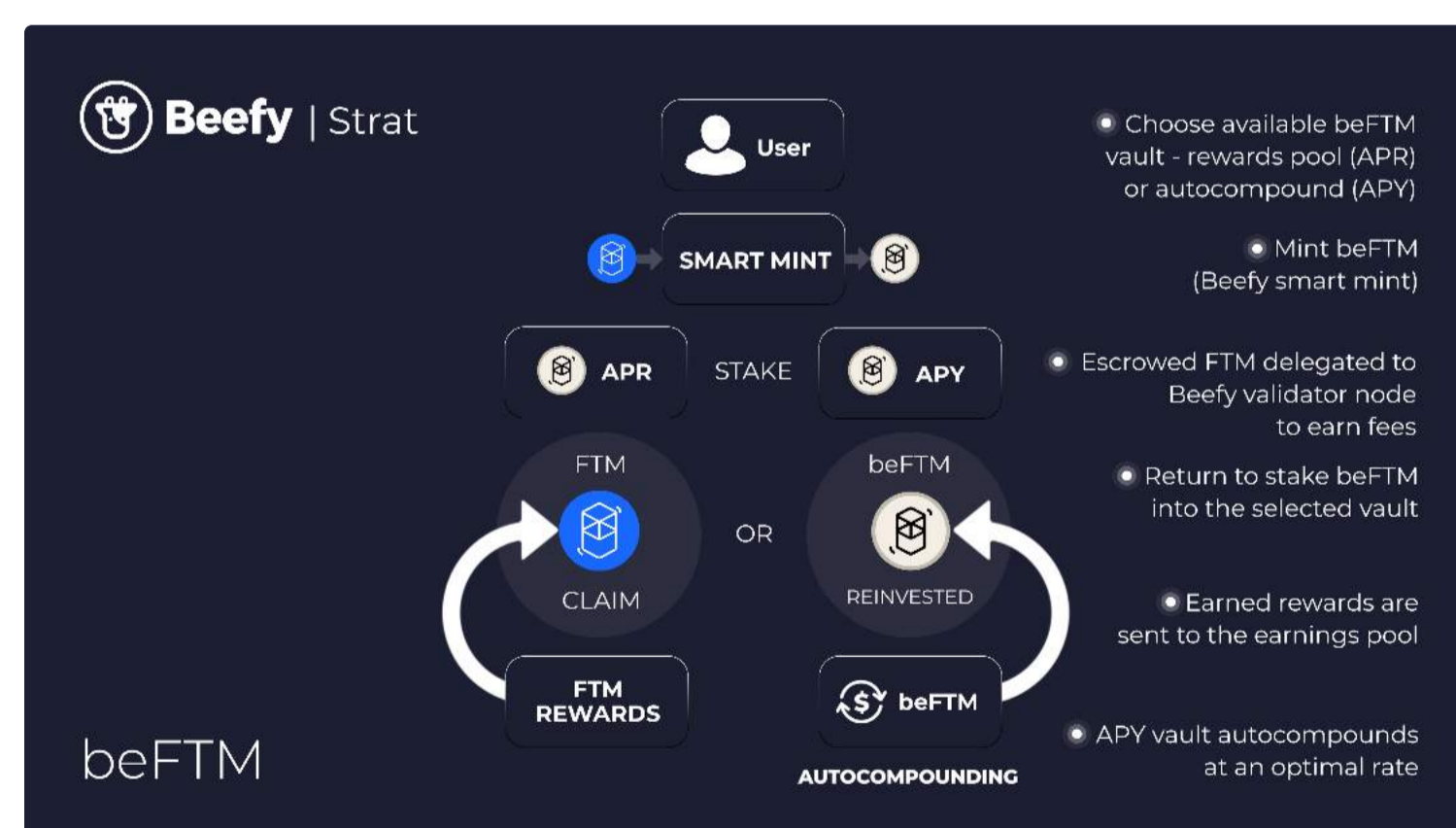
## beFTM Earnings Pool and beFTM Vault

After depositing FTM to get beFTM, the FTM gets delegated to Beefy's Validator Node. The validator stakes the FTM tokens to help secure the Fantom network, and earns WFTM tokens as a reward. The WFTM rewards will be collected and distributed daily to the beFTM Earnings Pool where users can stake beFTM to earn WFTM. The beFTM Vault deposits the beFTM in the Earnings Pool and uses the share of the vault to buyback beFTM with the WFTM rewards. By doing so, the beFTM vault adds a continuous buying pressure to the beFTM token price.

## What can I do with beFTM?

- Deposit beFTM in the beFTM Vault for compounded interest in the form of more beFTM;
- Deposit beFTM in the WFTM Earnings Pool for simple interest with WFTM rewards;
- Stake in beFTM-FTM liquidity pools on all the major DEXes on Fantom.

## How does the beFTM strategy work?



## How does beFTM keep its peg?

While beFTM is backed 1 to 1 by FTM, it is not designed to keep a 1 to 1 peg with FTM: ultimately its price determined by the market.

There are several mechanisms in place to help beFTM maintain peg against FTM. First of all, Beefy's Smart Minter will not issue new beFTM tokens when the price of beFTM is below the price of FTM, it will instead buy beFTM from a liquidity pool helping to restore the peg and giving the user a more profitable option. Secondly, the beFTM vault adds a continuous buying pressure to the beFTM token price. Lastly, arbitrageurs will either buy and sell beFTM depending on the peg.

In a hypothetical situation where the peg crashes to 0 with no signs of recovery, Beefy's Validator Node can be paused, forfeiting all pending rewards, and after waiting 1 full week everyone will be able to redeem their beFTM for the delegated FTM.

## V1 Deprecation

In late March 2023, the Beefy Core team proposed the deprecation of the existing version of beFTM, to allow all FTM deposited in the current version to be unlocked and an improved V2 to be developed and deployed. The [BIP-67] proposal called to initiate an unlock on 3 April 2023, with the full unlock being completed by 3 April 2024. The proposal also calls for a V2 to be developed closer to the time to include better redemption capabilities. The proposal passed with a 99% margin on 2 April 2023.

# binSPIRIT

Beefy-wrapped inSPIRIT

## What is SPIRIT?

SPIRIT is the native token of SpiritSwap, a decentralized exchange native to the Fantom blockchain. It rewards holders with a share of the platform's revenues and also acts as a governance token. SPIRIT has a fixed supply and decaying emissions model.

Users can stake and lock their SPIRIT tokens on SpiritSwap for a fixed period between 1 week and 4 years, to receive a proportional amount of inSPIRIT. inSPIRIT holders receive a range of benefits, including: (1) a proportion of the protocol's revenues (which vary week by week, but have been as high as 80% APR); (2) voting rights in the protocol's governance and vault incentives gauge; (3) up to 2.5x boosted rewards from SpiritSwap farms, depending on the balance of inSPIRIT held by the user; and (4) access to bribes for voting for third party-incentivised gauges.

inSPIRIT is non-transferrable and the amount held by a given user decreases steadily to 0 when the lock is over. Users also cannot liquidate or transfer their locked SPIRIT positions until the end of the time lock.

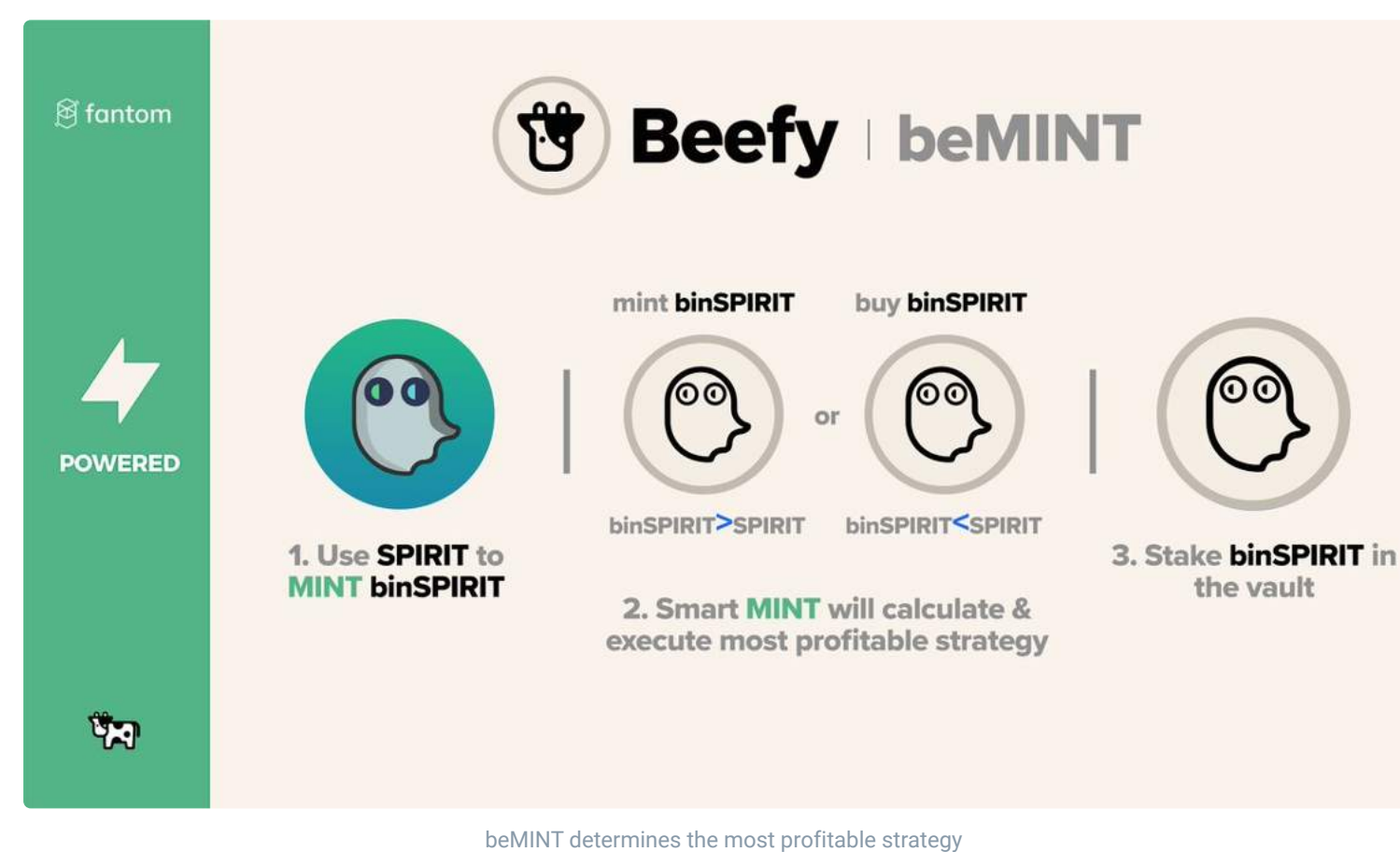
## What is binSPIRIT?

binSPIRIT is the Beefy-escrowed version of inSPIRIT which is staked to accumulate inSPIRIT. Holders of binSPIRIT receive each of the benefits of inSPIRIT detailed above, but without being constrained by the SpiritSwap lock mechanism.

The token is fully backed 1:1 by SPIRIT, though is perpetually locked for inSPIRIT, so cannot be withdrawn back to SPIRIT. However, binSPIRIT is a transferrable ERC-20 token, so can be swapped for other tokens on decentralised exchanges, allowing holders to exit their positions at any time.

## How does one get binSPIRIT?

Users can mint binSPIRIT on the [binSPIRIT vault page](#) at a 1:1 ratio. If it is more profitable to buy binSPIRIT, then Beefy's Smart Minter will do exactly that, yielding the user more binSPIRIT for their SPIRIT. Users can also get binSPIRIT by buying on a decentralised exchange.



## How does binSPIRIT work?

Beefy uses the deposited SPIRIT to accumulate as much inSPIRIT as it can by immediately and perpetually locking all deposits on SpiritSwap for the longest available period. It then puts the accumulated inSPIRIT to use to claim SpiritSwap protocol revenues, boost its SpiritSwap vaults and vote for SpiritSwap incentive emissions.

All protocol revenues are paid out in SPIRIT, which is automatically claimed by Beefy and then returned to the binSPIRIT vault, either by minting more binSPIRIT (if binSPIRIT is over peg) or by purchasing binSPIRIT on a decentralised exchange (if under peg) and then in either case redepositing the binSPIRIT into the vault.

All deposits, withdrawals and harvest rewards from Beefy's boosted SpiritSwap vaults are routed through the binSPIRIT contract, so that they may receive the benefit of its inSPIRIT boost. By keeping all of the accumulated inSPIRIT in one contract, and running all the vaults through that contract, binSPIRIT ensures that each vault receives the maximum available boost. All boosted rewards are retained by the individual Beefy vaults.

The binSPIRIT contract also submits votes on Beefy's behalf for SpiritSwap governance proposals and vault incentive gauges. binSPIRIT holders are empowered to [vote](#) on gauges, as described further below.

For further details of binSPIRIT's specific functionality and methods, see the description of its main [GaugeStaker smart contract](#).

## How can I earn with my binSPIRIT?

Once you're holding binSPIRIT, there are a few available options. You can either:

1. Stake it in the Beefy binSPIRIT vault to earn protocol revenue, which is autocompounded into more binSPIRIT;
2. Deposit it into a binSPIRIT liquidity pool with a decentralised exchange (e.g. SpiritSwap, Solidly) to earn trading fees (and potentially rewards); or
3. Deposit it into the SPIRIT-binSPIRIT LP on SpiritSwap, and stake in the [Beefy SPIRIT-binSPIRIT LP vault](#) to earn autocompounded trading fees and rewards.

## But what about fees?

Beefy strives to maintain some of the lowest yield-optimizing fees, and charges standard fees on its binSPIRIT vaults. No Beefy fees are charged for using Beefy's Smart Minter to convert SPIRIT into binSPIRIT.

## How does binSPIRIT keep its peg?

Should a user want to liquidate their position they will be able to trade binSPIRIT on an exchange. By contrast to inSPIRIT, users are never locked in and can exit their position at any time. This also means that binSPIRIT may not always be pegged to SPIRIT, but aims to maintain a loose peg.

If binSPIRIT goes under peg, then the contract will use the claimed SpiritSwap protocol revenues to buy back more binSPIRIT. This increases the size of each binSPIRIT holder's proportional share of deposited SPIRIT, leaving them with more than they would have achieved from locking the same SPIRIT directly on SpiritSwap.

If binSPIRIT goes over peg, then the contract will use protocol revenues to mint more binSPIRIT at a profit.

## Can I vote with binSPIRIT?

No. Though Beefy had previously set up a binSPIRIT voting system and a [dedicated Snapshot page](#), the move to SpiritSwap V2 has allowed Beefy to automate the process of bribing to help us obtain the highest return for holders. Beefy then returns all bribe earnings directly to the binSPIRIT vaults, facilitating high returns with minimal effort.

Though Beefy has automated the process, we are still open to receiving direct offers for binSPIRIT bribes. Please reach out to the Core team on Discord, Telegram or Twitter to find out more.

# beJOE

Beefy-escrowed JOE

beJOE was deprecated on the 6th of January, 2023. All beJOE can be burned 1 to 1 for JOE, and Beefy no longer takes any fees.

## What is JOE?

JOE is the native token of Trader Joe, a decentralized exchange native to the Avalanche blockchain. It rewards holders with a share of the platform's revenues and also acts as a governance token. JOE has a fixed supply and decaying emissions model.

Users can stake JOE to earn veJOE and receive boosted JOE rewards in selected Trader Joe Farms and governance voting power.

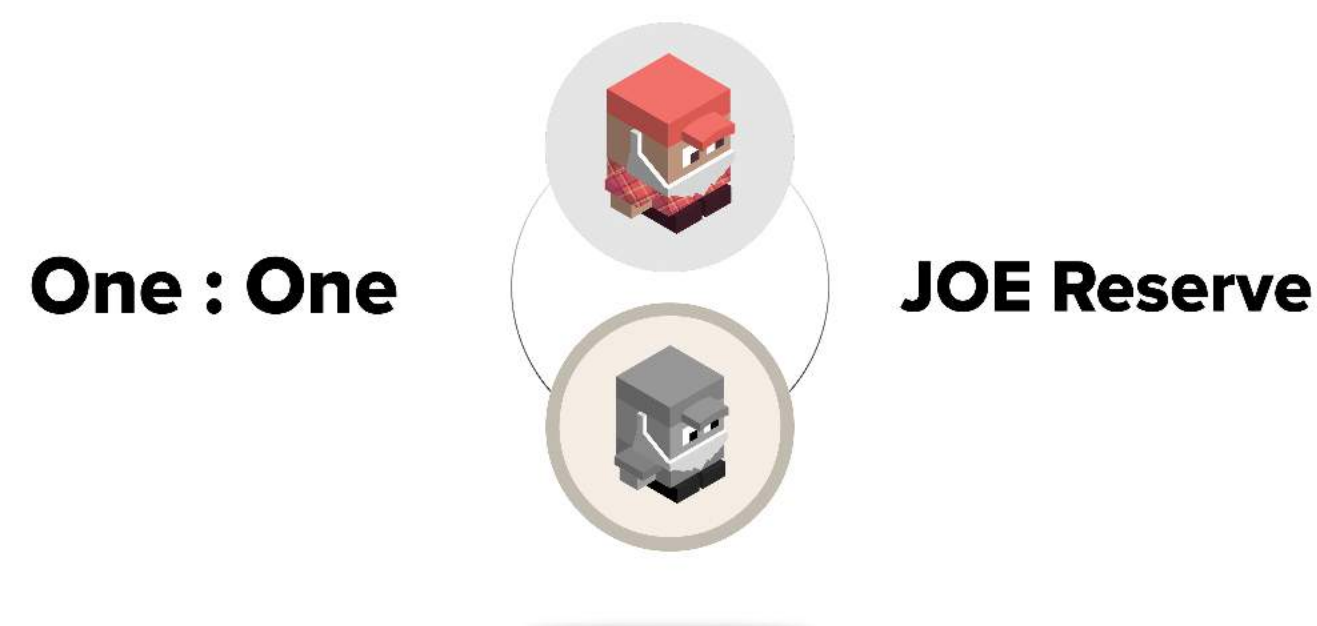
## What is beJOE?

beJOE is a Beefy-escrowed version of JOE staked to earn veJOE, maximizing emissions on boosted Beefy vaults. beJOE stakers earn 5% of emissions from those boosted vaults.

The token is fully backed 1:1 by JOE and can be redeemed for JOE held in reserve. This reserve only fills up when a new user deposits.

## How does one get beJOE?

You can mint beJOE on the beJOE vault or earnings pool pages at a 1:1 ratio. There is no incentivised liquidity for beJOE, instead there will be a withdrawal reserve.



beJOE is minted and burned at a 1:1 rate with JOE

## How does beJOE work?

When you mint beJOE, the contract will immediately try to stake the deposited JOE into veJOE, subject to two conditions: (1) the required reserve must be maintained; and (2) the staking must trigger a "Speed Up" bonus.

If the contract's JOE reserves at the time of minting exceed the required reserve amount (which is currently 20% of the contract's JOE already staked into veJOE), the contract can stake any excess JOE into veJOE. If the JOE reserves are under the required reserve amount, then the deposited JOE will be added to the reserve to cover the current shortfall.

The "Speed Up" bonus on Trader Joe doubles the rate of veJOE accruals for 14 days after JOE is staked, provided that the amount of JOE staked adds an additional 5% (or more) on top of the total amount of JOE already staked into veJOE. The contract therefore checks before staking whether the available balance (including both the deposit and any pre-existing excess over the required reserve) would meet or exceed the 5% threshold, and will only stake the available balance if it does. Otherwise, it waits for further JOE deposits to increase the available balance before staking.

Once the contract's JOE is staked into veJOE, it will automatically accrue veJOE rewards. veJOE provides 2 benefits: (1) boosted rewards on certain Trader Joe farms; and (2) (future) voting rights in Trader Joe governance. The amount of each benefit is proportional to the balance of veJOE held by the contract, so the aim is to accrue as much veJOE as possible.

Though veJOE rewards accrue constantly on the contract's staked JOE, the rewards must be harvested to be added to the contract's balance of veJOE. Rewards are automatically harvested whenever JOE is staked into veJOE, and otherwise manually harvested by the contract when new beJOE is minted but the conditions for staking are not met.

## How can I earn with my beJOE?

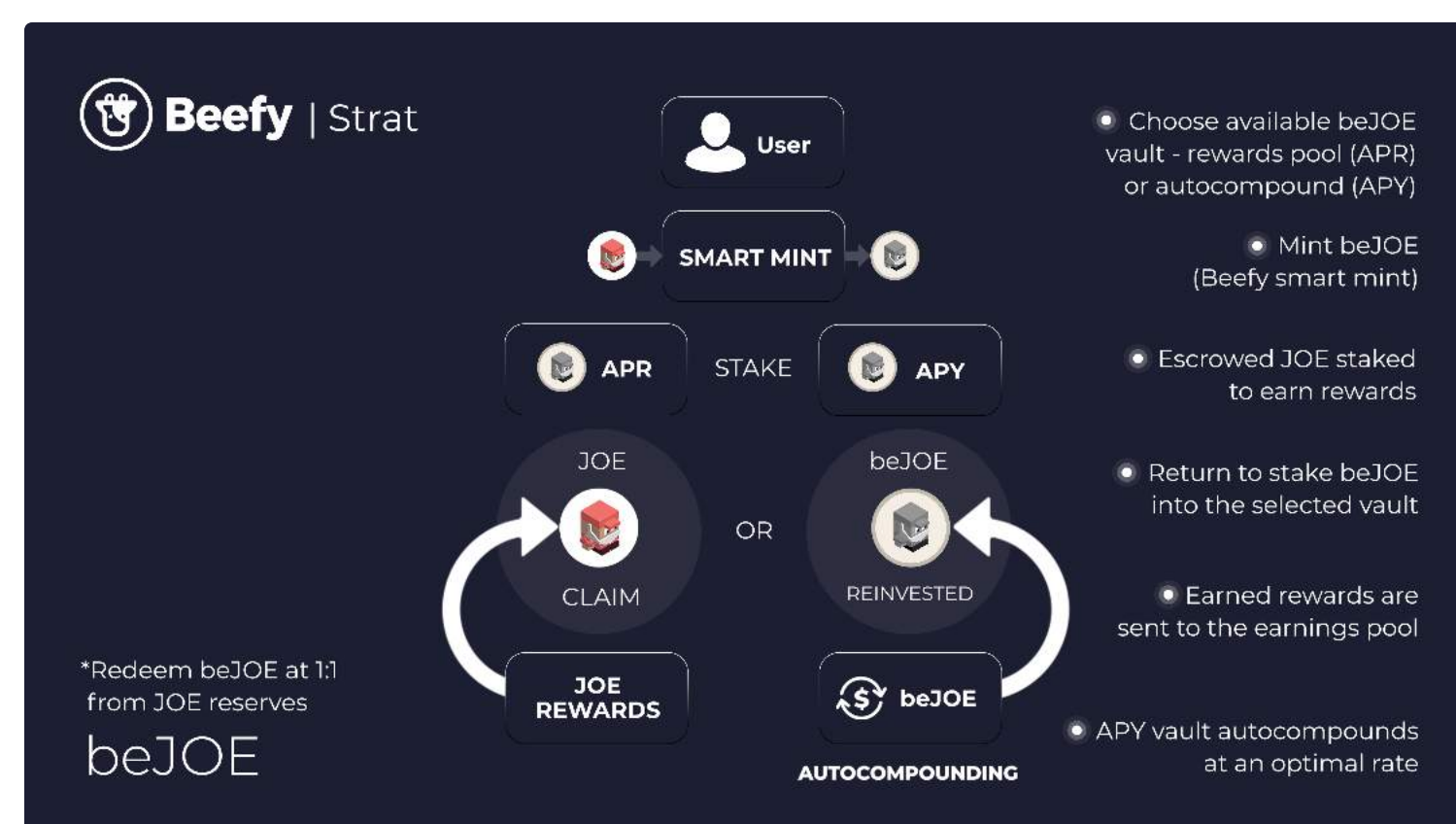
Once you're holding beJOE, there are a couple of available options. You can either:

1. Stake it in the beJOE vault to earn more beJOE; or
2. Stake it in the beJOE earnings pool to earn JOE.

In every Beefy vault for a Trader Joe farm which is boosted for veJOE holders, 5% of the value of every harvest will be diverted to the beJOE reward pool to be paid out to beJOE holders. This reflects the value added by the beJOE contract, which gathers the veJOE needed to boost these vaults.

As the boosted rewards on Trader Joe farms are paid out in JOE, these can be paid back to beJOE holders either directly in JOE (through the earnings pool), or by compounding beJOE (through the vault) by minting more beJOE with the JOE rewards.

## How does the beJOE strategy work?



## But what about fees?

Beefy strives to maintain some of the lowest yield-optimizing fees, and charges standard fees on its beJOE vaults, plus an additional 5% delivered directly to beJOE stakers.

## How does beJOE keep its peg?

There's no liquidity provided for beJOE, so it will always be at 1:1 with JOE. Users can burn beJOE for JOE while the reserve lasts. This reserve only fills up when a new user deposits.

In a last-resort scenario, all locked JOE in beJOE can be unlocked and released to the reserve. Beefy's competitors do not have this. However this would also end the boosts available for Beefy vaults and the rewards streamed to the beJOE reward pool, so is unlikely to be triggered.

## Can I vote with beJOE?

Not yet, but beJOE implements EIP-1271 (signature validation for contracts), so future governance/bribes with TraderJoe can be organised.

# beQi

Beefy-escrowed Qi

## What is Qi?

Qi is the native token of Mai Finance, a collateralised debt protocol native to the Polygon blockchain. Mai Finance is governed by QiDao, for which Qi is the governance token. Qi can also be used to participate in a share of the protocol's revenues. Qi has a fixed supply and decaying emissions model.

Users can stake Qi to earn eQi and receive an up to 4x boosted share of protocol revenues and governance voting power.

## What is beQi?

beQi is a Beefy-escrowed version of Qi staked to earn eQi, to boost the proportion of Mai Finance protocol revenues that Beefy earns and to participate in vault incentive gauge votes.

The token is fully backed 1:1 by Qi and can be redeemed for Qi held in reserve. This reserve fills up when new users deposit (if below the required reserve amount at the time), or the amount of reserve required gradually decreases as the contract's eQi gradually decreases and unlocks.

## How does one get beQi?

You can mint beQi on the beQi vault or earnings pools pages at a 1:1 ratio. There is no incentivised liquidity for beQi, instead there will be a withdrawal reserve.



## How does beQi work?

When you mint beQi, the contract will immediately try to stake and lock the deposited Qi into eQi, subject to the required reserve being maintained.

If the contract's Qi reserves at the time of minting exceed the required reserve amount (which is currently 20% of the contract's eQi), the contract can stake any excess Qi into eQi. If the Qi reserves are under the required reserve amount, then the deposited Qi will be added to the reserve to cover the current shortfall.

Once the contract's Qi is staked and locked into eQi, it receives two benefits: (1) boosted weekly protocol revenue rewards; and (2) voting rights in QiDao's governance and vault incentives gauge votes. Beefy's approach to voting for beQi is detailed below.

Mai Finance protocol revenues are paid out weekly, and currently include 100% of all farming rewards from the protocol's deposit fee revenue, which is used to farm Qi-MATIC, together with 30% of the debt repayment fees for all collateral types (plus a 25% weekly bonus). Individual rewards can be boosted by up to 4x where a user has the maximum amount of eQi (i.e. has locked their Qi for 4 years).

As the beQi contract perpetually re-locks its Qi deposits, it always strives for the maximum amount of boost. All of our received rewards are then distributed to our Beefy beQi vault and earnings pool for the benefit of beQi holders.

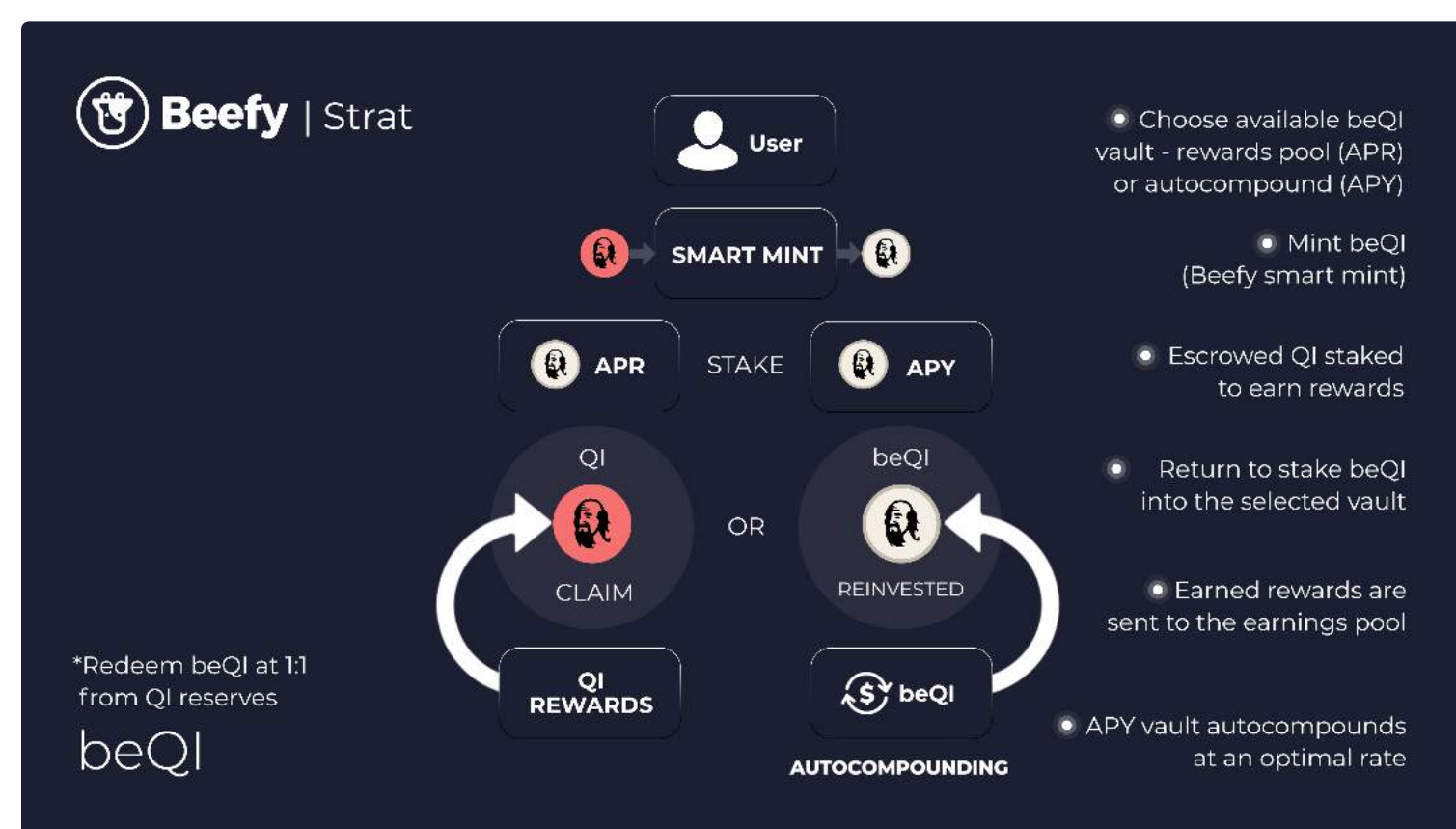
## How can I earn with my beQi?

Once you're holding beQi, there are a couple of available options. You can either:

1. Stake it in the beQi vault to earn more beQi; or
2. Stake it in the beQi earnings pool to earn Qi.

As eQi boosted protocol revenue is paid out in Qi tokens, these can be paid back to beQi holders either directly in Qi (through the earnings pool), or by compounding beQi (through the vault) by minting more beQi with the Qi rewards.

## How does the beQi strategy work?



## But what about fees?

Beefy strives to maintain some of the lowest yield-optimizing fees, and charges standard fees on its beQi vaults.

## How does beQi keep its peg?

There's no liquidity provided for beQi, so it will always be at 1:1 with Qi. Users can burn beQi for Qi while the reserve lasts without affecting the peg.

## How can I get my Qi back?

Whilst there are Qi tokens available in the reserve, you will be able to burn your existing beQi tokens (up to the amount of the reserve) to receive back an equivalent amount of Qi.

Where the reserve does not hold sufficient Qi tokens to facilitate the requested withdrawal, you will only be able to withdraw up to the amount of the reserve. Unfortunately, eQi locking also does not include an emergency release mechanism, so in this scenario holders will need to wait until the reserve is replenished.

As the required reserve amount is tied to the amount of eQi held by the contract, and all eQi balances reduce over time as the time left until unlock decreases, the amount of reserve required will also naturally decrease over time until the eQi lock is extended. As such, and assuming that no further Qi deposits are made to replenish the reserve, the required reserve amount will gradually decrease over time, meaning the amount of Qi available to withdraw will increase constantly.

## Can I vote with beQi?

Not at the moment. All Qi voting power is currently used by Beefy to vote in the weekly vault incentives gauge. Votes will typically be directed to Beefy collateral types (e.g. mooBIFI Fantom), which rewards our users who deposit their Beefy assets as collateral on Mai Finance.

Where our preferred collateral types are unlikely to qualify for the incentives gauge, we may also choose to accept bribes from external parties. The proceeds of all bribes are then distributed directly to our beQi holders. If you are interested in proposing a bribe to Beefy, please reach out to the Core team on Discord, Telegram or Twitter to find out more.

Beefy's Qi voting power can also be used for bribes, which are paid directly to holders



From June 2023, Velodrome has migrated to its V2 protocol, and \$VELO V1 has been retired in favour of \$VELO V2. To continue earning from \$beVELO, all users are advised to redeposit in the [beVELO Migration Pool](#), where earnings are paid out in \$VELO V2.

## What is \$VELO?

\$VELO is the native token of Velodrome Finance, a decentralised trading and liquidity marketplace native to the Optimism blockchain. It rewards holders with a share of the platform's revenues and also acts as a governance token for its weekly pool incentives gauge. \$VELO has a fixed supply and decaying emissions model.

Users can stake and lock their \$VELO tokens on Velodrome for a fixed period between 1 week and 4 years, to receive a vote escrow NFT ("veNFT"), which is used to record the amount of veVELO held by the user. Accumulating veVELO provides users with three benefits: (1) a share of the trading fees from swaps using the platform's liquidity pools; (2) the ability to direct \$VELO emissions distributed to platform liquidity providers; and (3) the opportunity to earn bribes from external parties by voting for their incentivised liquidity pools. \$VELO staking can be done through Velodrome's [web app](#).

veVELO is non-transferrable and the amount held by a given user decreases steadily to zero as the lock period moves towards its completion. Users also cannot liquidate or transfer their locked \$VELO positions until the end of the time lock.

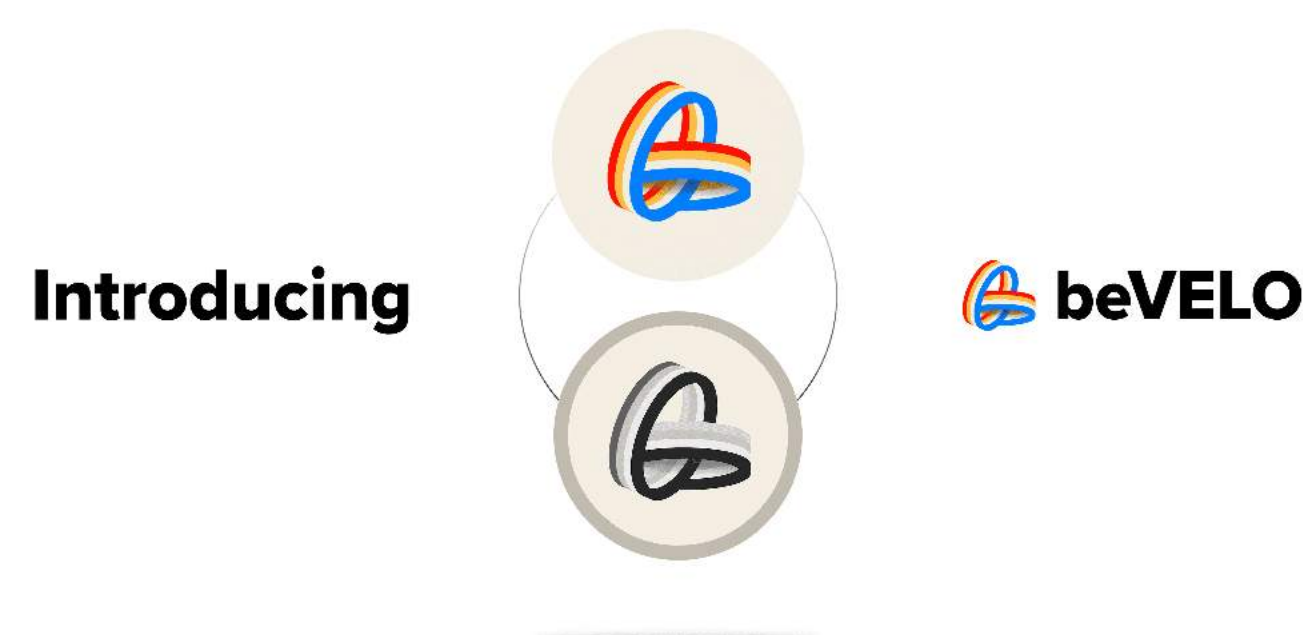
In June 2023, Velodrome migrated from \$VELO V1 to \$VELO V2.

## What is beVELO?

\$beVELO is a Beefy-escrowed version of \$VELO V1, staked for veVELO to take advantage of the various benefits offered to Velodrome stakers.

The token is fully backed 1:1 by \$VELO V1, which can be redeemed where tokens are available in the reserve. This reserve fills up on several circumstances:

- when new users deposit \$VELO into \$beVELO, if below the required reserve amount at the time; and
- if the contract's staked \$VELO V1 is left to gradually unlock.



beVELO is designed to capture the maximum possible rewards and benefits from Velodrome's vote escrow tokenomics.

## How does one get \$beVELO?

You can mint \$beVELO on the [\\$beVELO vault page](#) at a 1:1 ratio. There is also a Velodrome [liquidity pool](#) for swapping between \$beVELO and \$VELO V1, though in light of the [migration](#) to \$VELO V2, there is limited liquidity and a risk of high slippage when using this pool.

## How does \$beVELO work?

When you mint \$beVELO, the contract will immediately try to stake and lock the deposited \$VELO into veVELO, subject to the required reserve being maintained.

If the contract's \$VELO reserves at the time of minting exceed the required reserve amount (which is currently 20% of the contract's veVELO), the contract can stake any excess \$VELO into veVELO. If the \$VELO reserves are under the required reserve amount, then the deposited \$VELO will be added to the reserve to cover the current shortfall.

Once the contract's \$VELO is staked and locked into veVELO, it receives \$VELO V2 rewards paid out by Velodrome's protocol. As the \$beVELO contract perpetually re-locks its \$VELO V1 deposits, it always strives for the maximum amount of voting power and benefits. \$VELO V2 rewards are regularly harvested and returned to holders in the [beVELO Migration Pool](#).

## How can I earn with my \$beVELO?

Once you're holding \$beVELO, you can stake it in our [beVELO Migration Pool](#) to earn \$VELO V2. \$VELO V2 rewards are paid out directly rather than auto-compounded as the underlying \$VELO V1 has been retired.

\$beVELO can be deposited into our [beVELO Migration Pool](#) to earn \$VELO V2

## But what about fees?

Beefy strives to maintain some of the lowest yield-optimizing fees, and charges standard fees on its \$beVELO vault.

## Can I vote with my \$beVELO?

No. All \$VELO voting power will be used by Beefy to vote in the weekly liquidity pool incentives gauge.

Votes will typically be directed either to the liquidity pools offering the most in trading fees and bribes, or to the liquidity pools which support our \$BIFI token (e.g. BIFI-OP LP). Voting on Velodrome's incentivised liquidity pools takes place on its [web app](#).

If you are interested in proposing a bribe to Beefy, please reach out to the Core team on Discord, Telegram or Twitter to find out more.

## What's this I hear about migration?

In June 2023, Velodrome retired their original \$VELO V1 token as part of the upgrade to their V2 web app, and adopted the new \$VELO V2, which provides a range of additional features. As a result, Beefy has retired and redeployed many of the existing Beefy Velodrome vaults. Users can swap their V1 tokens for V2 at a 1:1 ratio on the [Velodrome app](#).

For \$beVELO, the new protocol design does not include functionality for \$VELO V1 liquid-staking derivatives, meaning the product cannot be upgraded. Instead, Beefy has implemented a [\\$beVELO Migration Pool](#), which allows users to get the benefit of its \$VELO V1 voting power, and rewards paid out in \$VELO V2. All users are urged to redeposit their \$beVELO into the Migration Pool to continue earning following the migration.

## What is OPX?

OPX is the native token of OPX Finance, a decentralised spot and perpetual exchange native to the Optimism blockchain. The OPX token is used to reward holders with a share of protocol revenues, whilst providing voting rights for use in protocol governance. OPX has adopted a fixed token supply model, though the current OPX token incorporates both minting and burning methods.

OPX offers a novel form of vote escrow tokenomics for its governance and profit share mechanics. This requires holders to stake and lock both their OPX tokens and an OPX NFT with the protocol. For this, holders receive veOPX, which reflects the holder's rights to participate in protocol governance and profits. OPX NFTs can only be obtained from an OPX mystery box (or third party sale) and are minted with a randomised level, which determines the percentage of boost applied to the holder's veOPX.

The key governance mechanism of veOPX is to decide on the distribution of OPX tokens among: (1) OPX liquidity (OLP) providers; (2) OPX stakers' profit share; (3) OPX protocol-owned liquidity; (4) OPX's developer treasury; (5) OPX's buy back and burn program; and (6) returning profit to DarkCrypto DAO - the creators of OPX. OPX [staking](#) and [voting](#) can be done through OPX's web app.

veOPX is non-transferrable and the amount held by a given user decreases steadily to zero as the lock period moves towards its completion. Users also cannot liquidate or transfer their locked OPX positions until the end of the time lock.

## What is beOPX?

beOPX is a Beefy-escrowed version of OPX staked for veOPX to take advantage of the various benefits offered to OPX stakers. As Beefy holds an OPX NFT of top-tier rarity (level 5), beOPX gives holders access to the maximum boosted profit share, which otherwise is reserved for the lucky few who acquire a similar rare NFT.

The token is fully backed 1:1 with OPX and can be redeemed for OPX held in reserve. This reserve fills up in several circumstances:

1. when new users deposit OPX into beOPX, if below the required reserve amount at the time;
2. when the contract harvests protocol revenues from OPX, if below the required reserve amount at the time; or
3. if the contract's staked OPX is left to gradually unlock.

## Introducing



 **beOPX**

beOPX is designed to capture the maximum possible rewards and benefits from OPX's vote escrow tokenomics.

## How does one get beOPX?

You can mint beOPX on the beOPX vault page at a 1:1 ratio. There is no incentivised liquidity for beOPX, instead there will be a withdrawal reserve.

## How does beOPX work?

When you mint beOPX, the contract will immediately try to stake and lock the deposited OPX into veOPX, subject to the required reserve being maintained.

If the contract's OPX reserves at the time of minting exceed the required reserve amount (which is currently 30% of the contract's veOPX), the contract can stake any excess OPX into veOPX. If the OPX reserves are under the required reserve amount, then the deposited OPX will be added to the reserve to cover the current shortfall.

Once the contract's OPX is staked and locked into veOPX, it receives a proportion of OPX protocol revenues which have been allocated for stakers, together with the ability to vote on future distributions of protocol revenue among the different available options.

As the beOPX contract perpetually re-locks its OPX deposits, it always strives for the maximum amount of voting power and protocol revenues. Earned revenues are regularly harvested, swapped for beOPX and redeposited to the beOPX vault to autocompound the return for beOPX stakers.

## How can I earn with my beOPX?

Once you're holding beOPX, you can stake it in our beOPX vault to earn more beOPX.

Where our beOPX contract earns protocol revenues by deploying its veOPX on the protocol, those protocol revenues are swapped back to beOPX and redeposited into the beOPX vault to give rise to an autocompounding effect. This maximises the yield for holders above what they could obtain alone from the protocol.

## But what about fees?

Beefy strives to maintain some of the lowest yield-optimizing fees, and charges standard fees on its beOPX vault.

## How does beOPX keep its peg?

There's no liquidity provided for beOPX, so it will always be at 1:1 with OPX. Users can burn beOPX for OPX while the reserve lasts without affecting the peg.

## How can I get my OPX back?

Whilst there are OPX tokens available in the reserve, you will be able to burn your existing beOPX tokens (up to the amount of the reserve) to receive back an equivalent amount of OPX.

Where the reserve does not hold sufficient OPX tokens to facilitate the requested withdrawal, you will only be able to withdraw up to the amount of the reserve. Users can then wait until the next harvest of protocol revenues, which will refill the reserve and allow them to burn their beOPX. Otherwise, OPX's locking mechanism does not include an emergency release mechanism.

As the required reserve amount is tied to the amount of veOPX held by the contract, and all veOPX balances reduce over time as the time left until unlock decreases, the amount of reserve required will also naturally decrease over time until the veOPX lock is extended. As such, and assuming that no further OPX deposits are made to replenish the reserve, the required reserve amount will gradually decrease over time, meaning the amount of OPX available to withdraw will increase constantly.

## Can I vote with my beOPX?

No. All OPX voting power will be used by Beefy to vote in the protocol revenue distribution votes. Voting takes place on OPX's [web app](#).

# GMX and GLP

⋮

Last Update: November 2022



## What is GMX?

GMX is a multi-chain decentralized exchange on Arbitrum One and Avalanche, letting you easily trade perpetual futures. On the GMX DEX, you can trade BTC, ETH, AVAX, LINK, and UNI with up to 30x leverage from your self-custodial crypto wallet.

As a trader, you enter and exit your positions with zero price impact, while profiting from minimal spreads and deep liquidity. Reliable price data is realized by aggregating price feeds, so users also benefit from lowered liquidation risks.

## What is GLP?

GLP is a token minted to liquidity providers on GMX. GLP represents an index of assets, used for leverage trading and swaps on the GMX platform. You can mint it using any of the index assets and burn it to redeem any of the index assets. The cost of minting and redeeming is calculated based on:

*(the total worth of assets in the index including profits and losses of open positions) / (GLP supply)*

After minting GLP, it's automatically staked to earn escrowed GMX (esGMX, an escrowed version of GMX's utility and governance token), multiplier points, and ETH or AVAX rewards depending on the network.

## How does Beefy's GLP strategy work?

Beefy is creating two new Vaults, each one taking GLP deposits on either Arbitrum or Avalanche. To mint GLP, you'll have to deposit an asset contained in the GLP index on GMX as outlined previously. GLP is not bridgeable between Arbitrum and Avalanche.

The strategy works as follows:

1. Users stake GLP in one of the two Beefy Vaults: [The Arbitrum GLP Vault](#) and [Avalanche GLP Vault](#).
2. Beefy claims and stakes all the earned esGMX and multiplier points.
3. The earned esGMX is never vested but instead used to boost earnings for more native tokens.
4. The earned multiplier points also boost earnings for more native tokens.
5. Beefy claims fees and mints additional GLP to also earn more fees.

## What are the benefits of the strategy?

GLP holders provide liquidity for leverage traders and profit when leverage traders make a loss. If leverage traders profit, GLP holders make a loss. In a long-term crab market, these Vaults are essentially a bet against traders while also taking a stable index position.

## The GLP transfer cooldown

There is a 15-minute cooldown between minting and transferring GLP. Depositing GLP to the Beefy Vault will only succeed when the cooldown period on the user's account has expired. The cooldown also affects withdrawals from the Beefy Vault as every harvest will mint new GLP to the Vault's strategy address.

Withdrawals will therefore only work 15 minutes after the most recent harvest. This will be displayed on the Vault UI. At the contract level, Beefy has introduced safeguards to prevent withdrawal griefing.

## The Beefy GMX Referral Link

Traders who use our [GMX link](#) will get a 5% discount and grant Beefy treasury addresses a 5% rebate.

# General

## Is Beefy audited?

Our first auditor was DefiYield, which audited \$BIFI token, the RewardPool and all the timelocks.

Beefy is also audited by Certik, which guarantees the robustness of our smart contracts and the safety of funds invested through Beefy.

Certik has audited some of the most complex and reusable investment strategies used within the platform. This ensures the safety and sturdiness of important smart contract aspects that the majority of our users interact with.

All Beefy audits can be found [here](#).

## What is a yield optimizer?

A yield optimizer is an automated service that seeks to gain the maximum possible return on crypto-investments, much more efficiently than attempting to maximize yield through manual means.

Each vault has its own unique strategy for farming, which normally involves the reinvestment of crypto assets staked in liquidity pools. At the most simple level, it farms the rewards given from staked assets and reinvests them back into the liquidity pool. This compounds the amount of interest received and increases the amount staked that the yield is based on. A yield optimizer can repeat this up to process up to thousands of times a day.

This fairly simple method is the principle reason behind the large APYs found on Beefy. Compounding fees are amortized among all vault participants, making it cheaper for the user.

## What's the difference between APR and APY?

APR (Annual Percentage Rate) is the yearly interest, minus fees. This does not include compounding effects that occur from reinvesting profits. If you were to invest \$100 with 100% APR, you would make \$100 in profit in a year time.

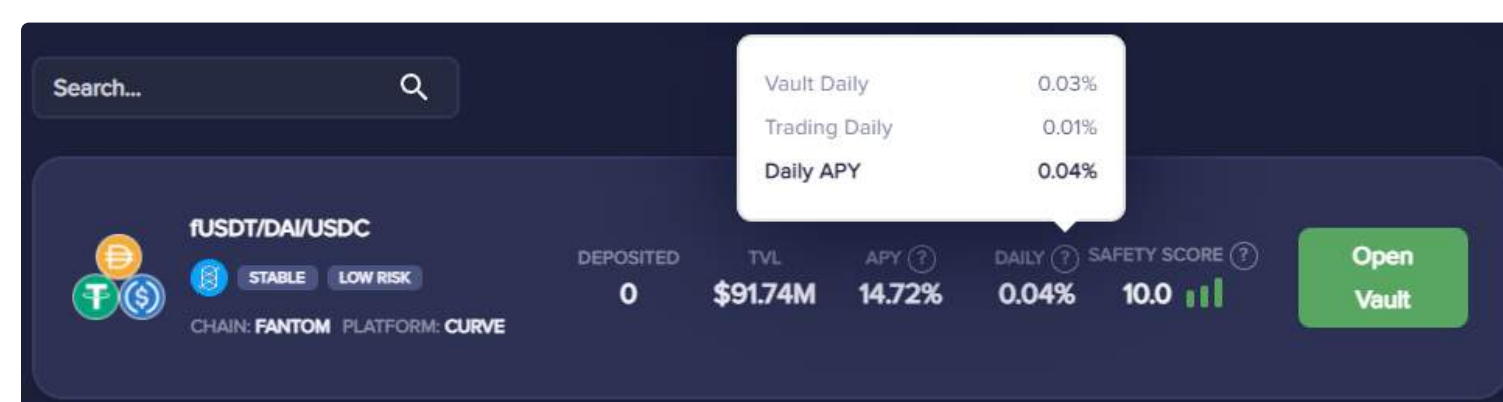
If you however reinvest your profits regularly, you will compound your interest. This calculated over a year gives you your APY (Annual Percentage Yield). The more often you compound your interest, the greater the difference between APR and APY.

## How does APY work?

APY is the annual percentage yield offered from a particular investment. This takes into account compound interest, giving you an accurate idea of your returns compared to simple interest.

Large APYs in the percentage of thousands are possible with investments that provide daily yields of 1% or more. Due to your liquidity pool rewards being constantly farmed and reinvested, the interest compounds on larger and larger amounts.

## What do Vault Daily and Trading Daily mean?



Trading Daily means how much your liquidity tokens will increase in value. Liquidity pools share trading fees amongst all liquidity providers, as introduced by the [Uniswap liquidity model](#). Trading Daily is affected by trading volume and the percentage of swap fees allocated to liquidity providers.

Vault Daily means how much your token will increase in number. Due to the vault constantly farming rewards, and reinvesting that, your deposited token amount will increase. Vault Daily is affected by the yield farm rewards (i.e. additional incentives besides trading fees), such as CAKE on Pancakeswap.

Trading Daily and Vault Daily can be multiplied by 365 to compute Trading APR and Vault APR. Vault APR is then converted to Vault APY to factor in compound interest. The displayed total APY percentage is calculated as follows:

$$APY = (1 + vaultAPY) * (1 + tradingAPR) - 1$$

**i** To calculate the Trading APR, Beefy uses on-chain data and a 24 hour period to determine the trading volume and subsequent fees, whereas most DEXes use a 7 day period. This may lead to differences in the displayed APY when compared with a DEX, but know that it is due to the calculation method. In fact, we argue that Beefy is more accurate because it uses a shorter time span which reflects changes in Trading APR sooner.

A handy tool to convert APR to APY is: [APRtoAPY.com](#)

## How do I contribute to Beefy?

Beefy is a community powered project from day one. If you want to join the ever growing pool of contributors, it depends what you would like to work on. We have open places for Solidity devs, or devs wanting to start a career in Solidity, to join as strategist and deploy vaults (and earn passive income from the strategist fees). Beefy is on a lot of chains and there are often opportunities for simple and complex vaults. You can start with simple ones and then progress to the harder ones as your knowledge of Solidity grows. You don't have to be the best right at the start, and rest assured that there is a rigorous review process in place to ensure safety and quality. You can reach out to our lead strategists in [Beefy's Discord](#) in #strategy-devs.

Beefy would also like people to work on non-strategy projects; pretty much anything you can think of can be formulated into a grant. Speak with others in the cowmoonity about projects and join one of the teams or lead one up yourself, you can be paid for any work you do to make Beefy better. A quick list of previous grants: [here](#) and [here](#). Beefy V2 is an ongoing project that requires all kinds of devs, not just technical ones; design input is crucial to improve the UI/UX.

Beefy's [GitHub](#) embraces the idea of open collaboration, hence many of the repositories are open-source. We use CONTRIBUTING.md files to allow people to just make contributions or recommendations by means of Pull-Requests. You can get started even just by updating the Git docs or fixing a typo, it helps you get closer to the team of contributors.

If you have an interest in business development you could help with partnerships and proposing business decisions to the DAO. Beefy is still a relatively new business that can use talented people to help advise the core team.

There is marketing that you can contribute to too, if you can write a decent tweet then you can help out in #tweet-development. The [Discord](#) has a #social-watch channel where links to Beefy mentions on social media are posted, you can help out with user queries there or in the [Discord](#) or [Telegram](#) itself. Moderators of Discord and Telegram are (variably) paid positions too and are usually the first line of customer support.

The best way to get involved is to just go ahead and get started, help where you can, contribute to discussions and collaborate with everyone.

## What is the difference between a Vault and an Earnings Pool?

In a Vault you earn more of what you deposited into it, with compound interest (APY). In an Earnings Pool you earn a different token than the asset you deposited, with linear interest (APR).

An example is the BIFI Maxi Vault, in which you earn more BIFI exponentially, and the many BIFI Earnings Pools, in which you earn linear interest in the form of \$ETH, \$BNB, \$AVAX and more.

## Why does it cost so much gas to deposit into a Beefy vault?

Many of Beefy's vaults "Harvest on Deposit". This means that when you deposit into the vault, you are also calling the harvest function. Calling the Harvest function is more complex than a simple deposit and thus has a higher gas limit/fee. Beefy does this so that it is impossible for malicious actors to steal yield so a withdrawal fee is not required. This greatly benefits long-term investors since the withdraw fee can be removed.

Almost all of the vaults on more inexpensive chains like Fantom and Polygon harvest on deposit. You can also tell if a vault harvests on deposit if there is no withdrawal fee.

As the Harvest Caller, you will also receive some of the wrapped native chain token in as a reward for calling the harvest. See [Beefy Fees Breakdown](#) for more information on the Harvest Caller.

## How can I find out how much earnings I have accumulated?

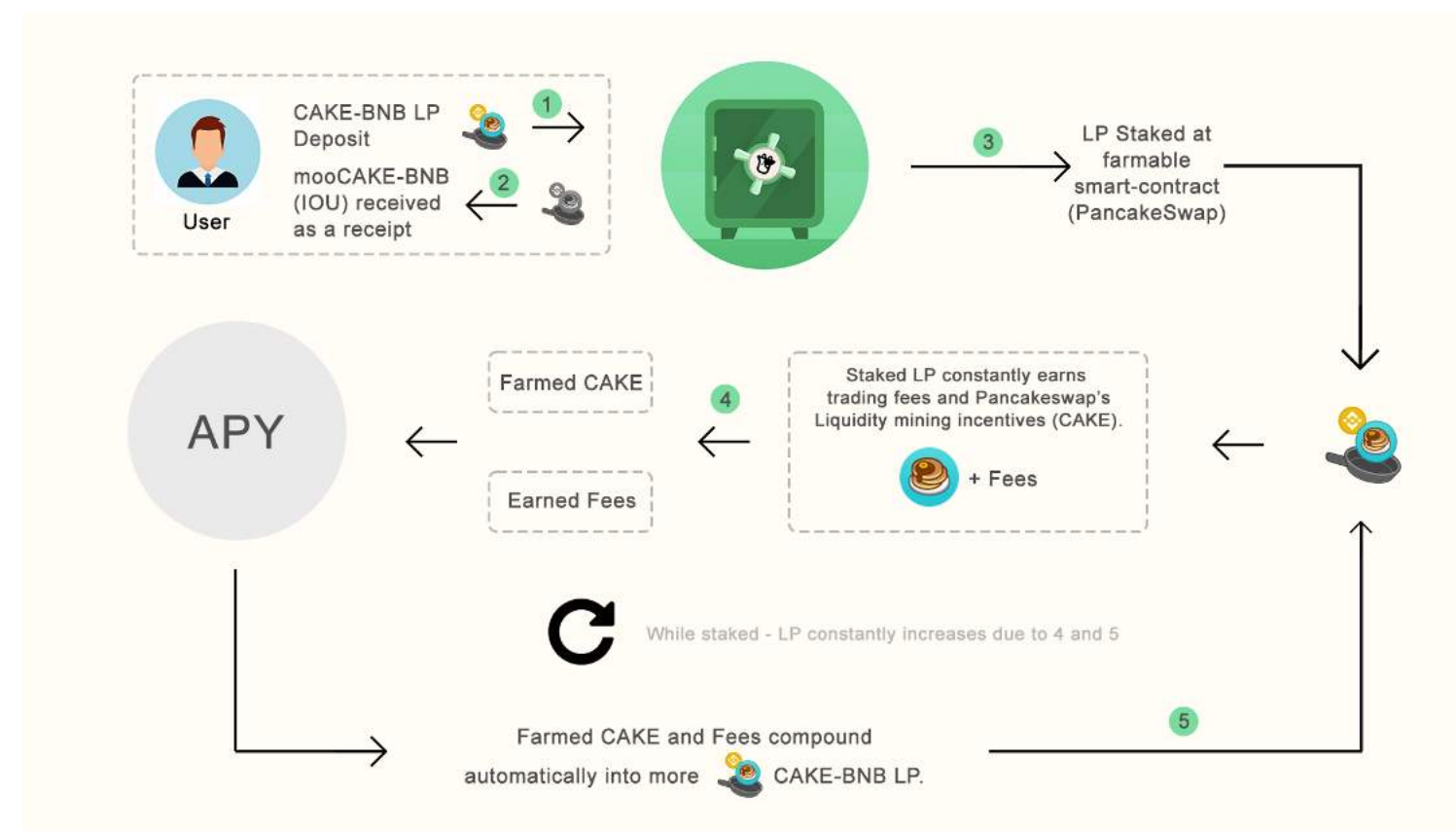
Your rewards are added to your deposited token amount on each harvest and compound cycle. You can use a DeFi dashboard that will be able to calculate exactly how much profit you have made on your investments. External tools such as [TopDeFi](#) will read your wallet address and give you an accurate picture of your initial investment and current earnings.

# Infographics

Explaining Beefy using simple infographics

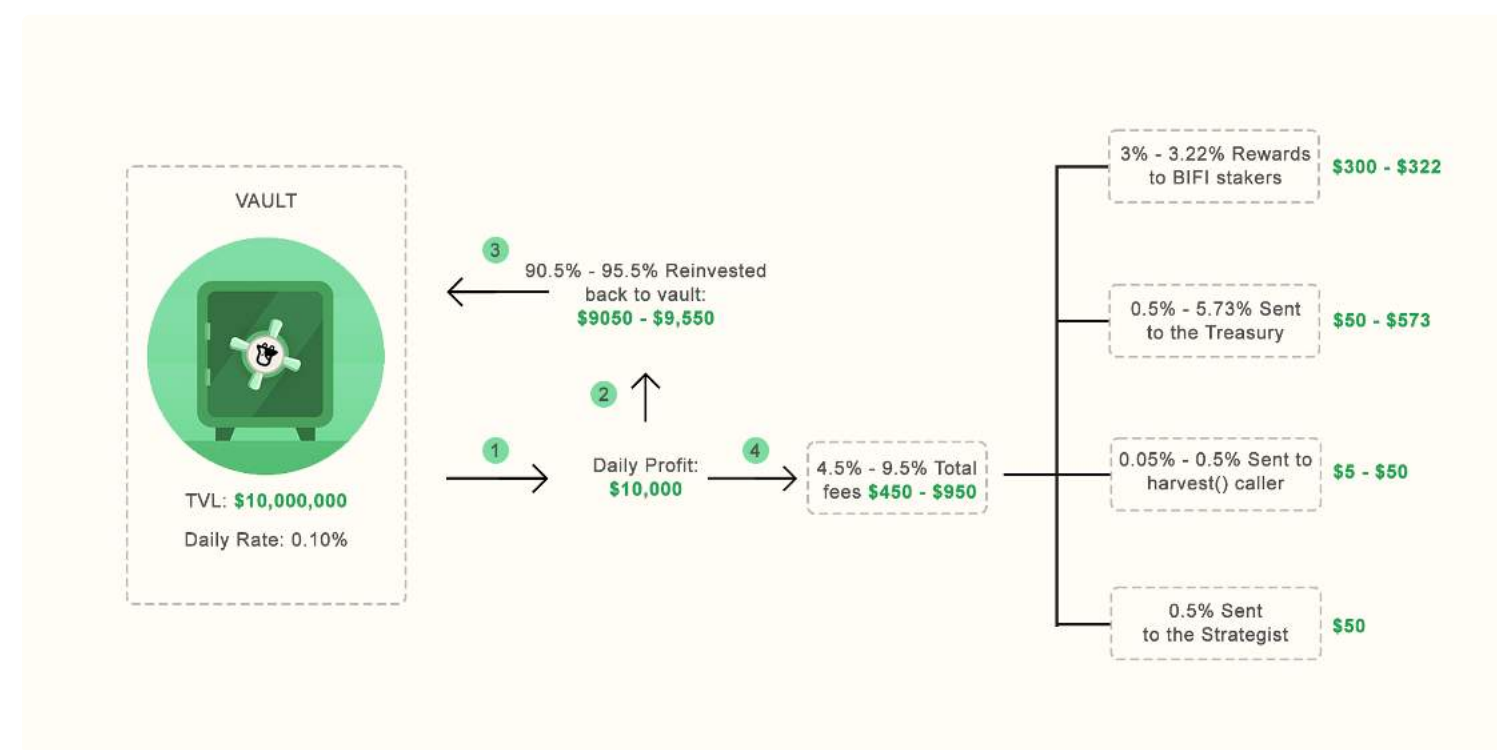
This page helps to explain various key aspects of Beefy's vaults, tooling and protocol using easy to understand infographics and short descriptions. Though the services we offer are advanced and highly technical, we agree with the adage that *"if you can't explain it simply, you don't understand it well enough"*.

## Vault Yield Farming



At Beefy 'you earn what you stake', regardless if this is a liquidity pool (LP) token or a single asset. In this example, staking CAKE-BNB LP will result in more CAKE-BNB LP over time. This effectively grows your share in the liquidity pool and thus allows for more and more rewards over time. All of this with Beefy doing the required work, while you can sit back and relax!

## Vault Fee Structure

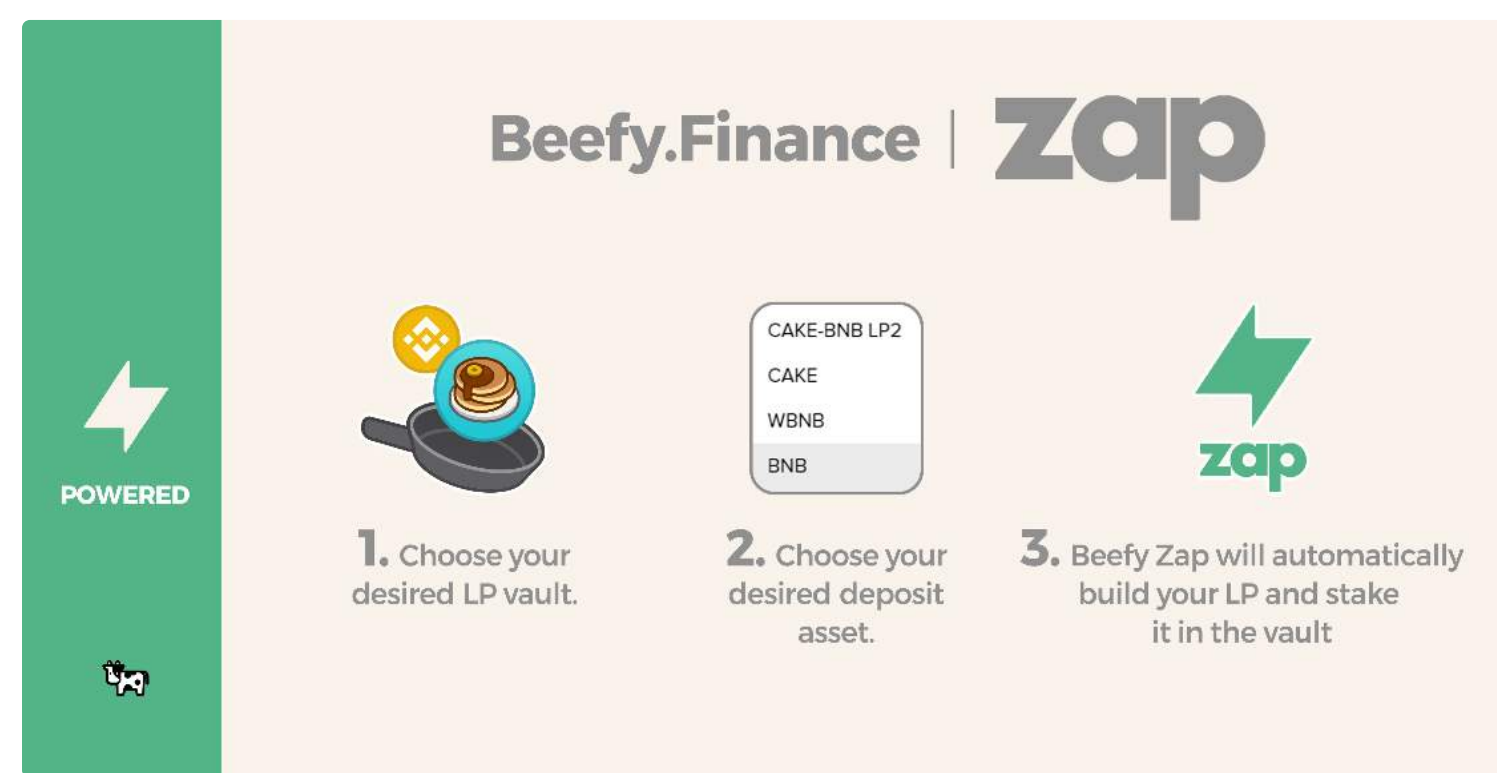


\*What you see is what you get\*: the fees are already accounted for in the displayed APY!

More on the vault fees [here](#).

## Beefy ZAP

Our Beefy ZAP tooling automatically builds the deposit (or withdraw) positions you need for our Beefy vaults from other assets, thereby saving you the time, energy and cost of obtaining the necessary assets and building the necessary liquidity positions yourself. Here's a guide on [How to Use Beefy ZAP](#).



ZAP V1 allows users to enter supported LP vaults with any of the base assets in the LP.

Our ZAP V1 tool automatically builds liquidity pool (LP) tokens from any of the base deposit assets, such as from BNB for the CAKE-BNB LP2 shown above. When the time has come that you want to withdraw from a LP vault, ZAP V1 also supports withdrawing back into any base asset, for instance you can exit into CAKE instead of BNB. This saves you the hassle of manually adding and removing liquidity at a yield farm.



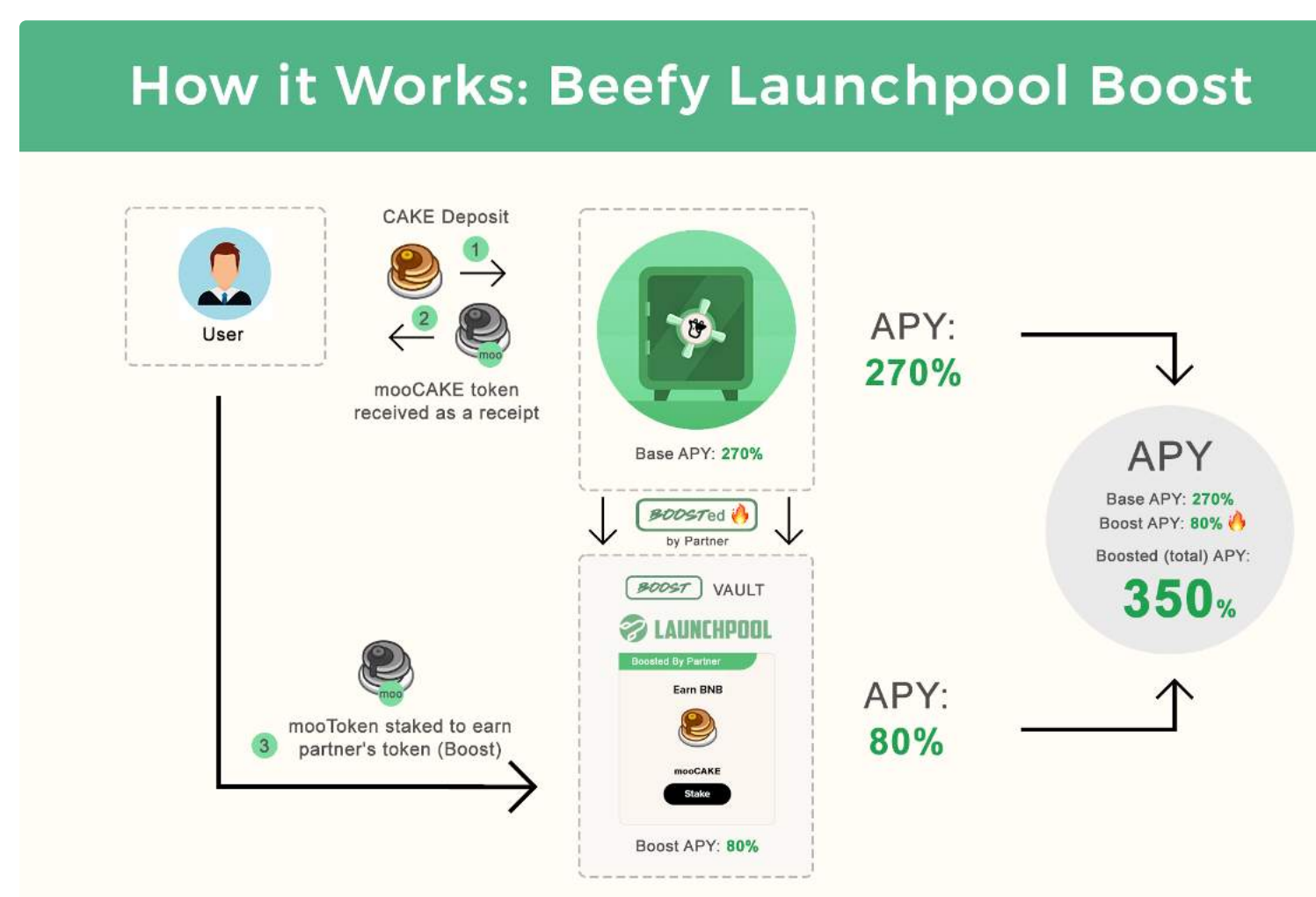
ZAP V2 goes one step further, so users can enter supported vaults from any of a selection of blue chip, native or stablecoin assets, regardless of whether the asset forms part of the vault's asset base.

The ZAP V2 tool takes things one step further, using the power of DEX aggregators like 1inch to add initial swaps to the process, so users can move from common blue chip tokens (e.g. WBTC, ETH), native tokens (e.g. MATIC, BNB) or stablecoins (e.g. USDC, USDT, DAI) into the underlying tokens needed for the liquidity position. That way, you can access our market-leading returns on new products without handling anything but the tokens and stables you already have in your wallet.

The ZAP quote that's displayed during the deposit or withdraw process already has the ZAP fee taken into account. Additionally, the ZAP fee is only deducted from token swaps. The fees first accumulate in a batch treasury, and after some time are swapped to stables and sent to [Beefy's Treasury](#). The original Beefy ZAP V1 remains free to use.

When using ZAP, always check your quote! While ZAP does protect you against market slippage (price changes at the time of order and time of fulfillment), it does **not** protect you against price impact (how much your transaction will change the price of the tokens in the liquidity pool). As a general rule, the smaller the liquidity of the asset, the larger the risk of price impact.

## Beefy Launchpool Boost



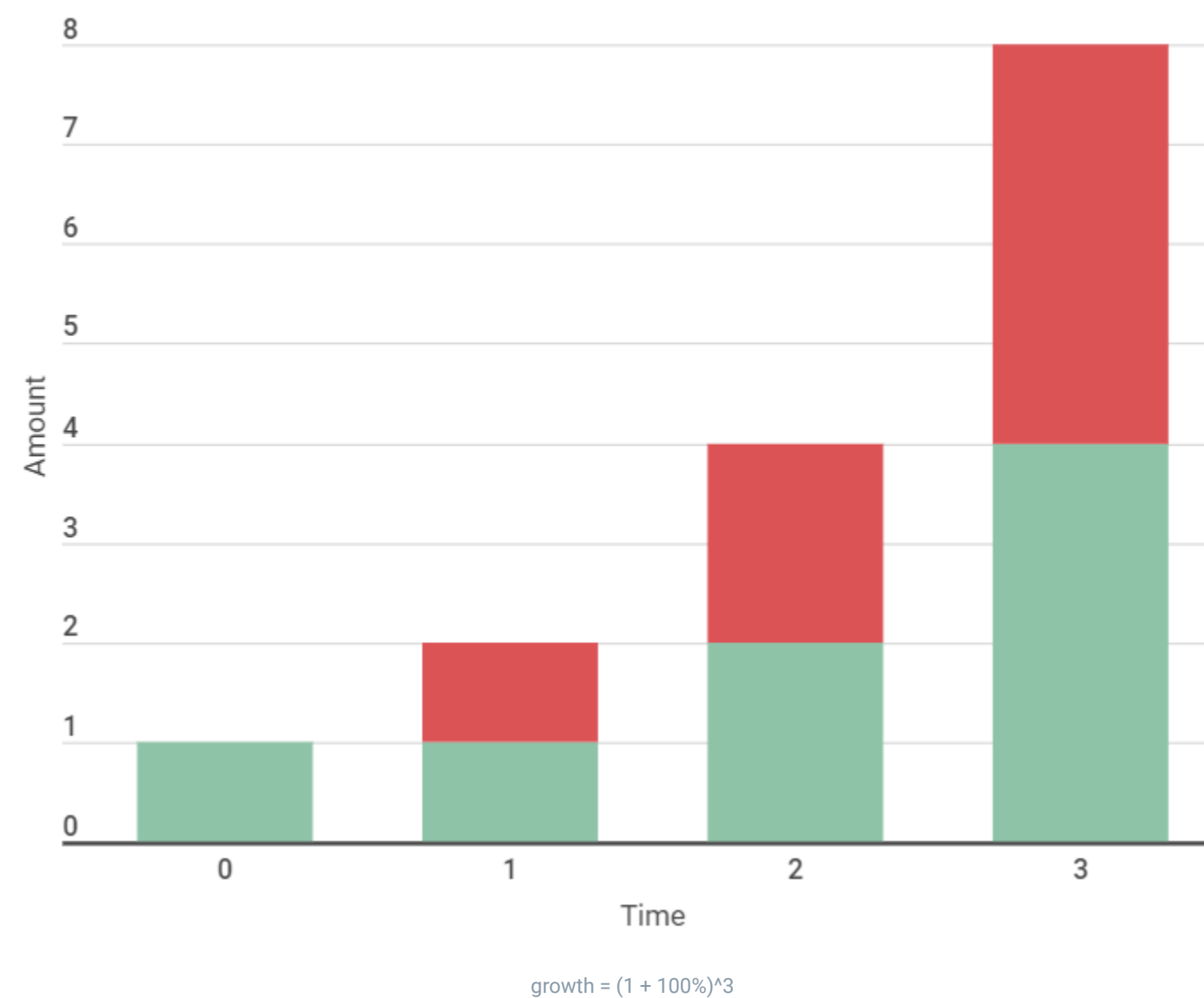
When a vault gets boosted in Beefy's Boost, you earn both the base asset and the partner's token! For more info, read the [Boost FAQ](#) [here](#).

## What is the APY?

Let's look at a typical yield farm, where they state an APY (annual percentage yield) as +100% for example. The traditional definition of APY **is the real rate of return earned on an investment taking into account the effect of compounding earnings**. Using this terminology would indicate that the yield farm was compounding earnings for you. That is simply not the case. A more applicable terminology to use would be APR (annual percentage rate), meaning the annual rate earned through an investment. By definition this would mean that your 100% yield farm would double your original investment at the end of year 1 without reinvesting any earnings. But what about if you reinvested that entire amount the next year and the year after that?

## Understanding Exponential Growth (Compounding)

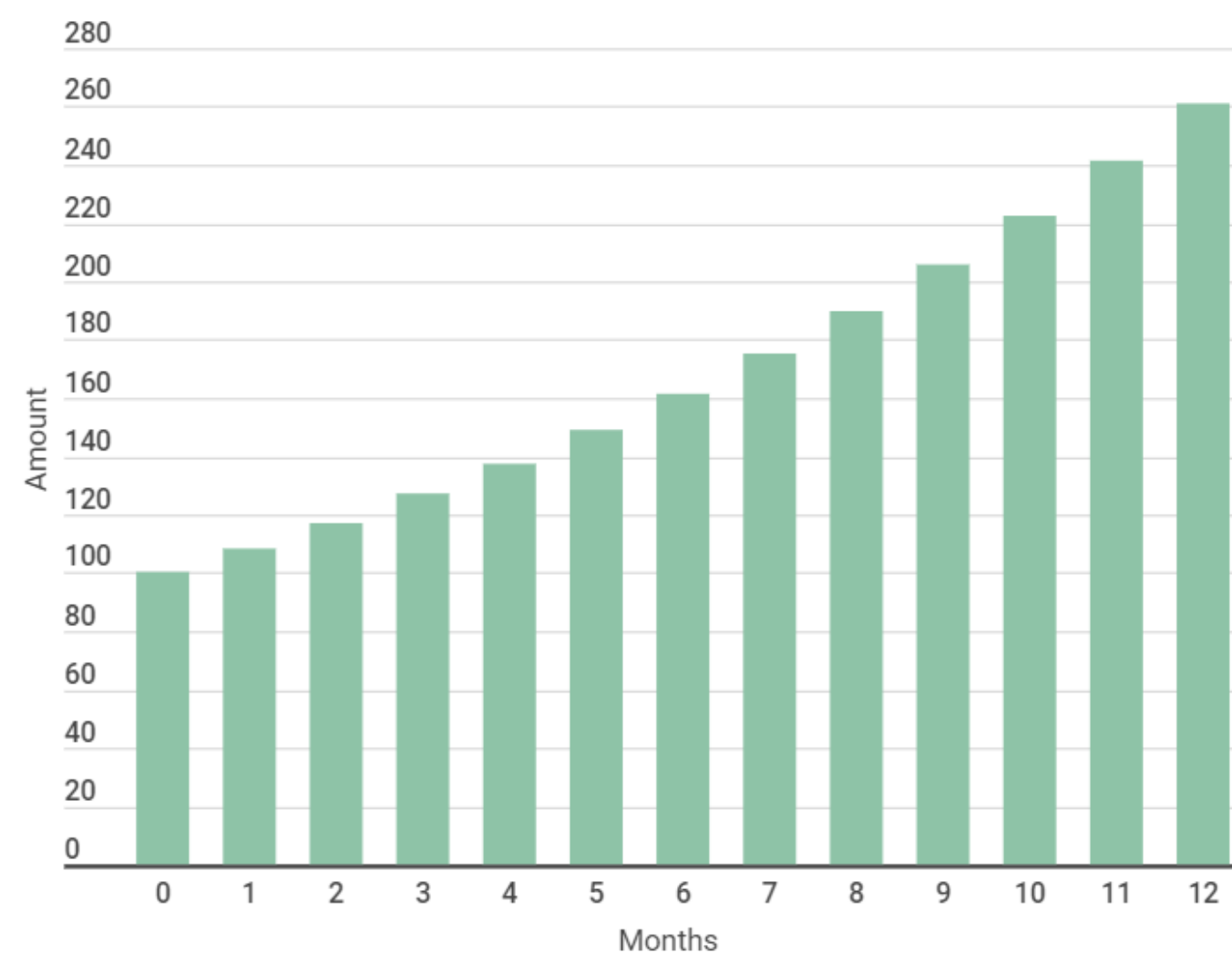
Growth whose rate becomes ever more rapid in proportion to the growing total number or size. The simple formula for this is **growth = (1 + r)^x**, where 'r' = return and 'x' = number of 'times'. For example, your money doubles every year if you get 100% yearly return. After 3 years you would have 8x your original investment.



## Ok, so what REALLY is the APY?

A typical investment does not just pay out on a yearly basis, but in smaller terms (ie: daily, monthly, etc). For yield farming, returns are even paid out on a per block basis. With an average of 28,800 blocks a day and cheap transaction fees this can allow for a significant amount of exponential growth or compounding of your return. Let's look at how to break that down...

- Compound =  $P * (1+r/n)^{nt}$  Example :  $100 * (1+1/12)^{(12*1)}$
- P = principal or starting balance
- r = APR = 100%
- n = compounding periods = 12 months
- t = time = 1 year
- The simple APY calculation in excel can also be stated as =EFFECT(r, n)



Year 1 end would be 261 tokens or 161% APY versus 100% APR w/o compounding

# How to deposit in a Vault

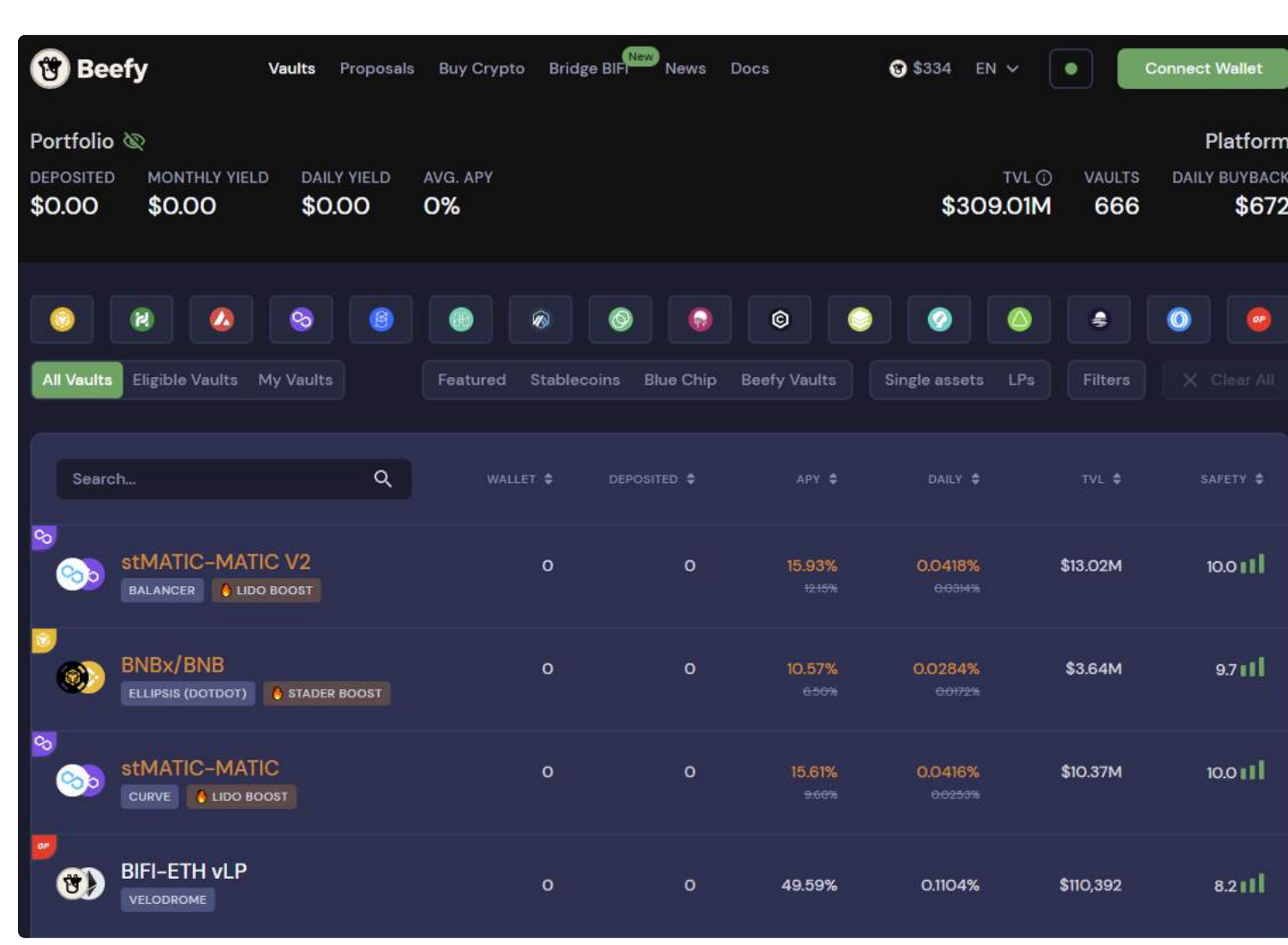
This visual guide will walk you through every step in depositing funds in a Vault

## Prerequisites

- You must have the vault's underlying token(s) in your wallet. See here how to fund your wallet:  
[Funding your wallet](#)
- You must use a supported wallet, such as Metamask or Trustwallet:  
[Connecting your wallet to Beefy.](#)

## Walkthrough

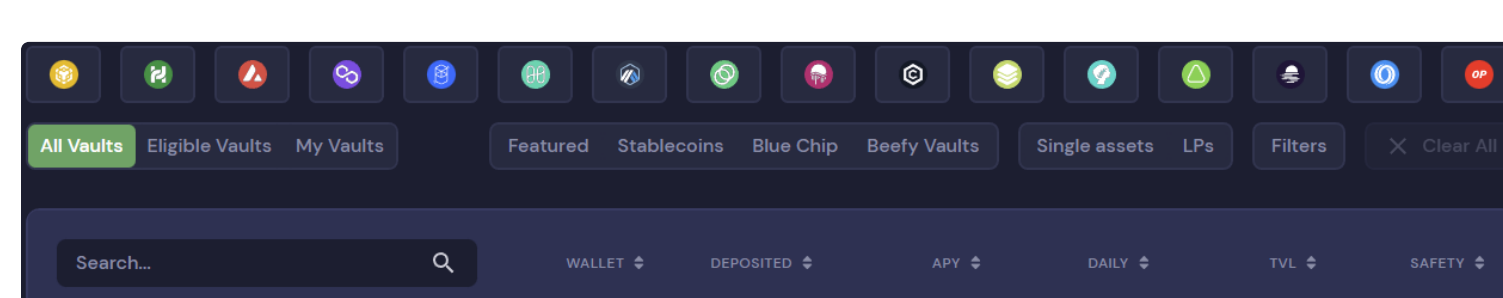
### 1. Go to the Beefy app page:



Screenshot date: 10 October 2022

Again, make sure that your wallet is connected and that it is funded with tokens.

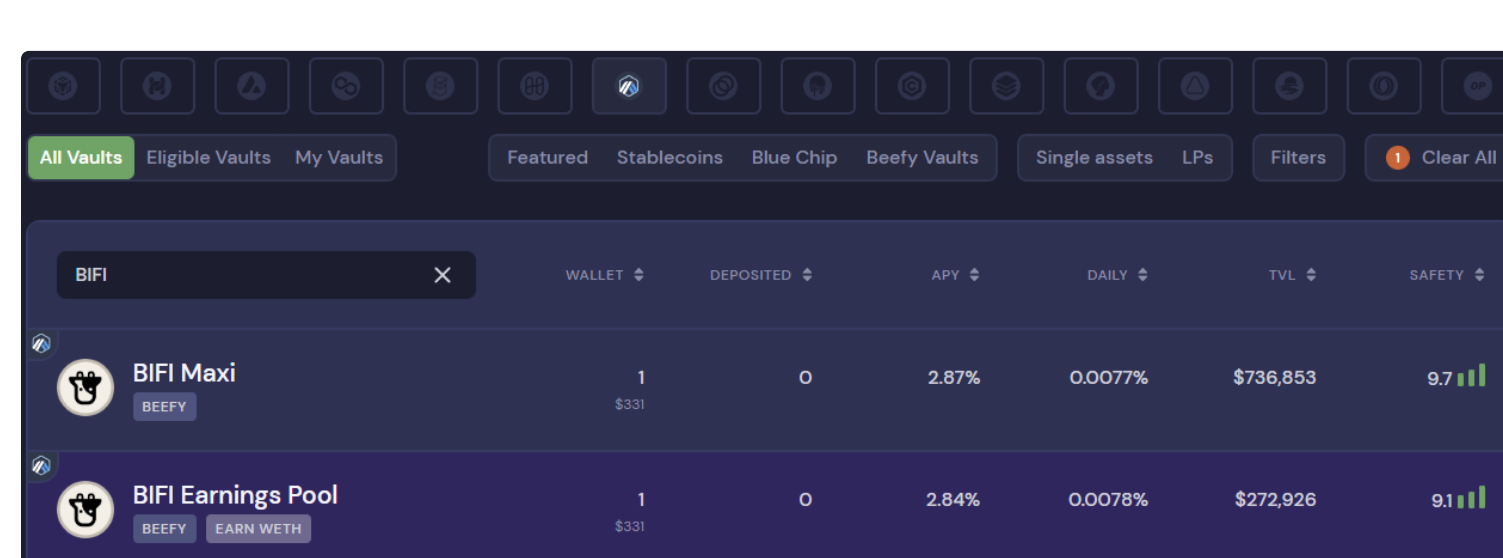
### 2. Use the filters to find a vault you want to deposit into:



The blockchain logos, the preset selection buttons and the search field all act as filters.

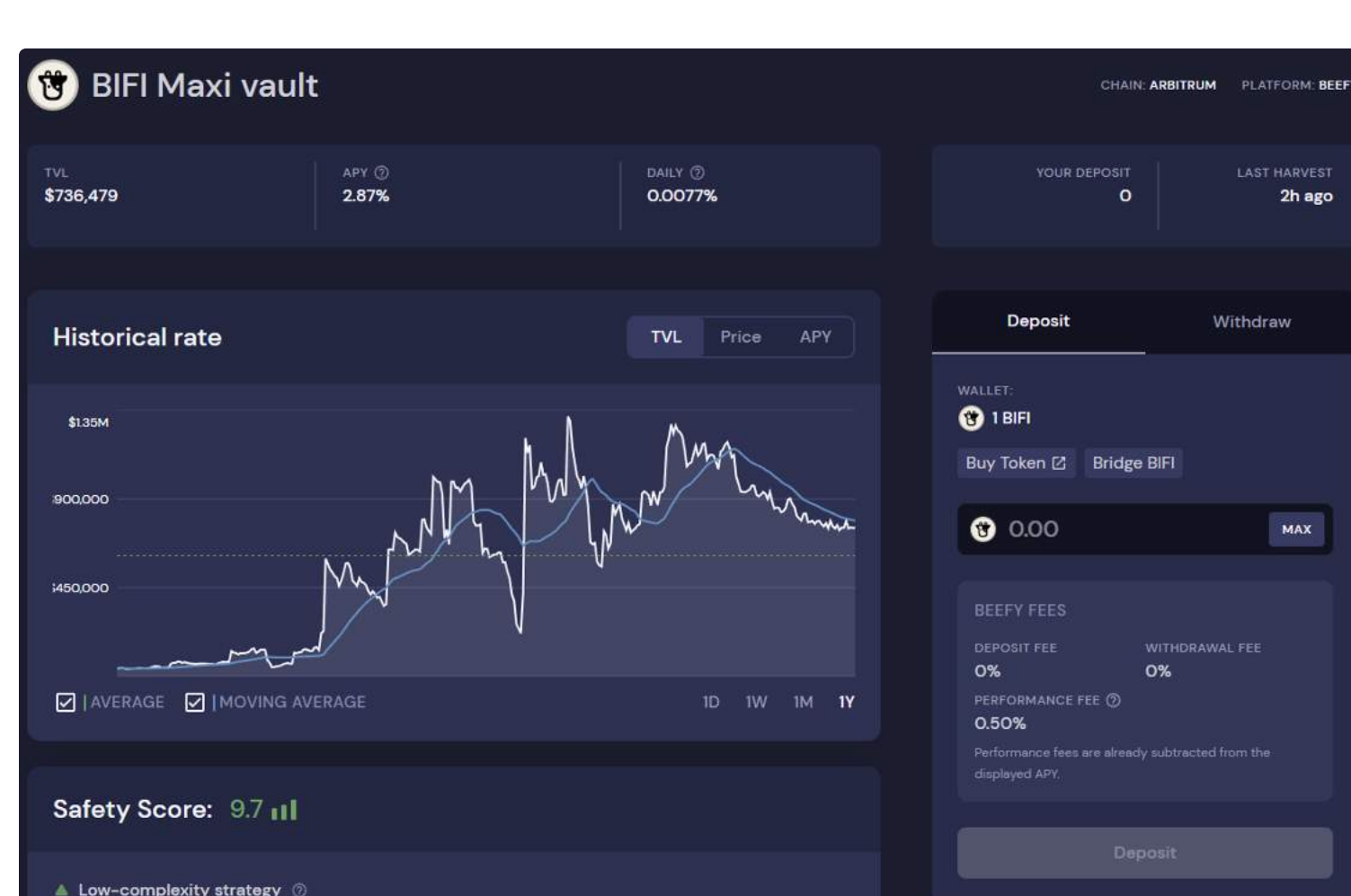
### 3. Example of filter usage

In this guide, we will use the BIFI Maxi vault on Arbitrum as an example:



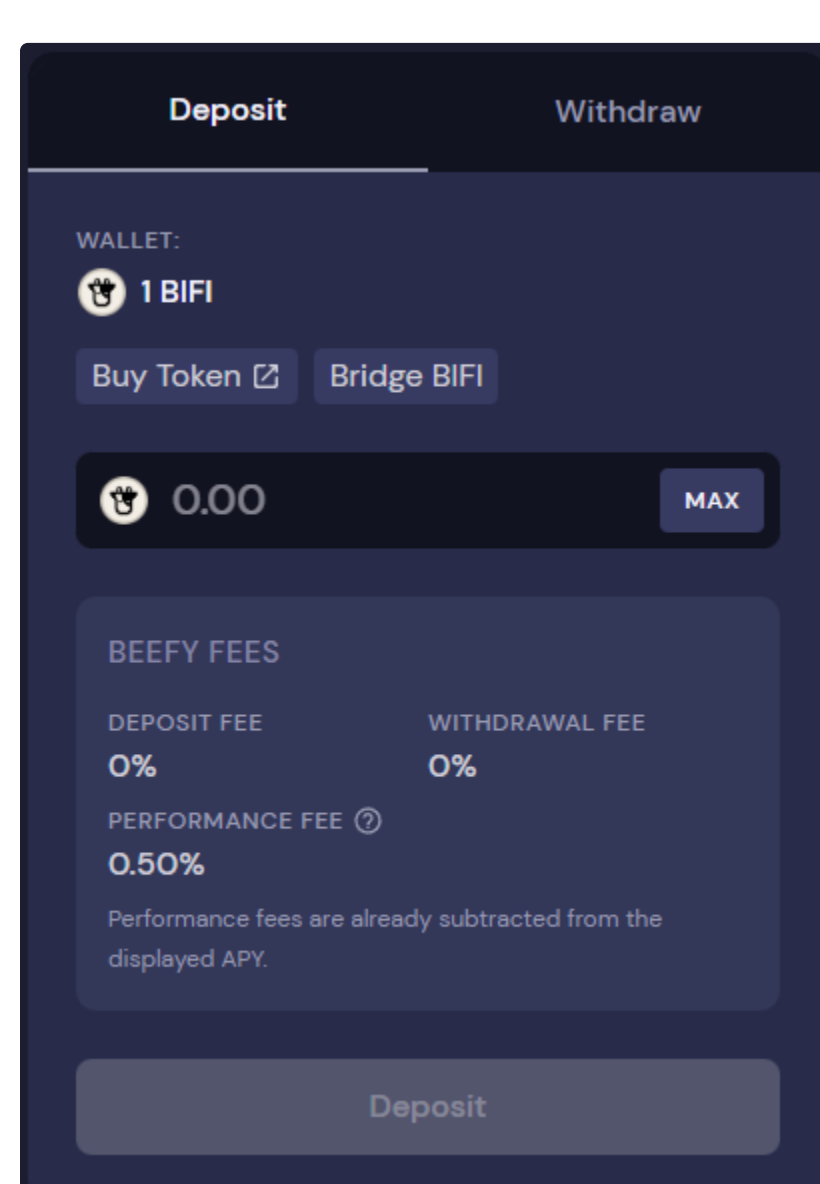
Note that the BIFI Earnings Pool, in which you can earn WETH by depositing BIFI, also shows up. Open the BIFI Maxi vault by clicking anywhere in the field above.

### 4. Inside the BIFI Maxi vault:



There is a lot of information inside the vault, such as TVL (Total Value Locked), Price and APY (Annual Percentage Yield) historical rates, the [Beefy Safety Score](#), Token Asset details, and the Vault's compounding strategy ([# what-is-a-vault-strategy](#)).

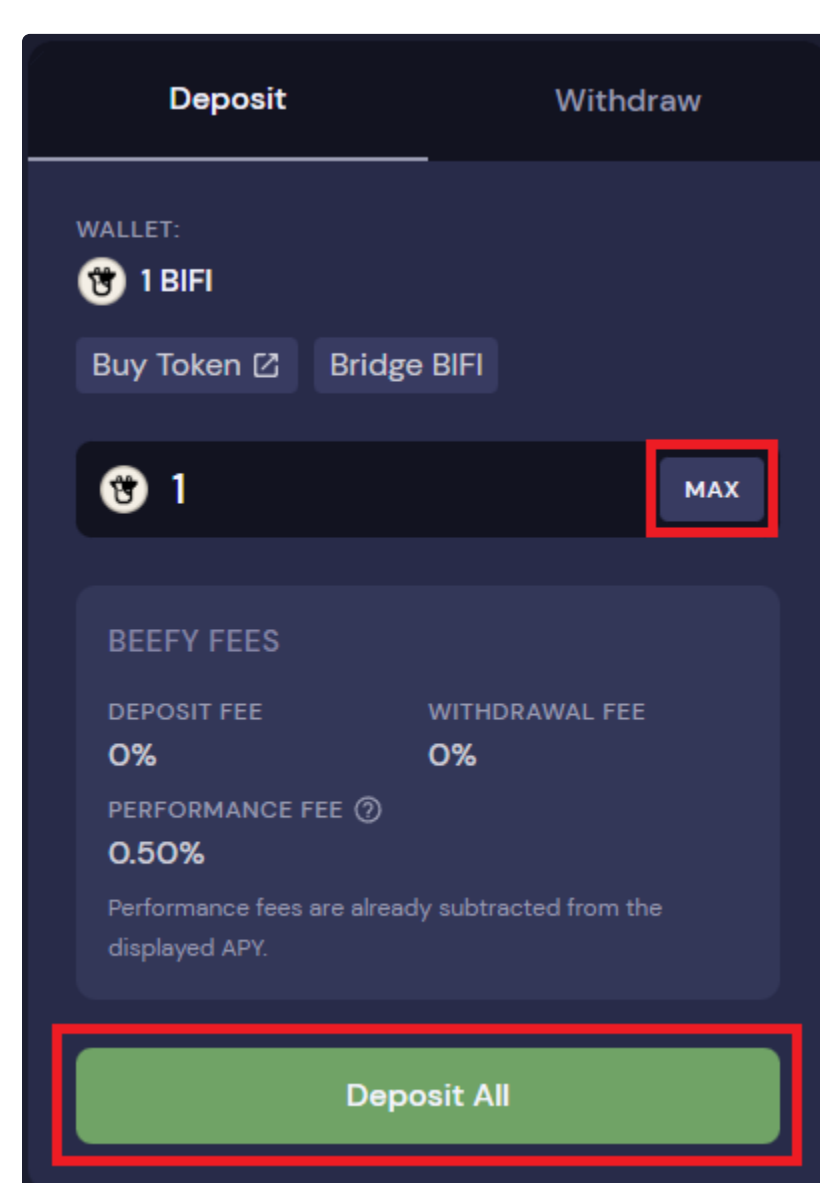
### 5. The Deposit and Withdraw module:



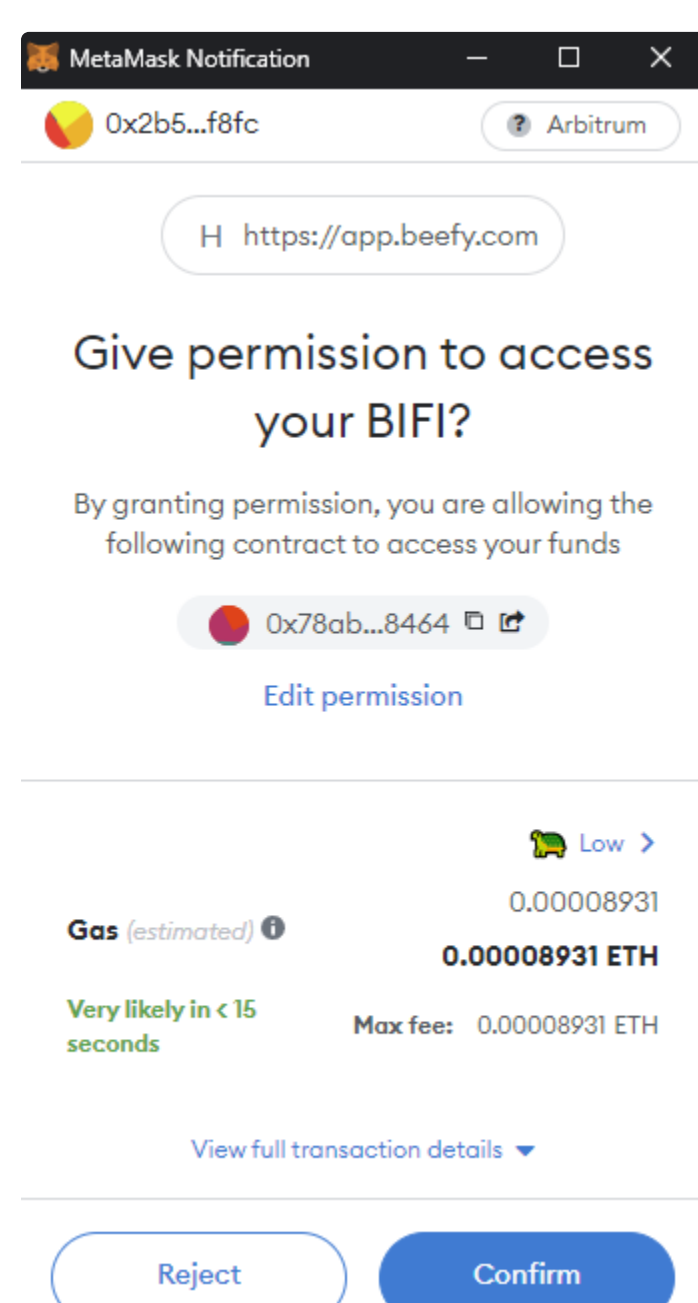
The vault already sees we have 1 BIFI available in our wallet to deposit. There is a "Buy Token" link provided in case you do not have any BIFI or wish to buy more BIFI to deposit, as well as a "Bridge BIFI" button to bridge BIFI from another blockchain to Arbitrum ([How to bridge BIFI cross-chain](#)). A deposit field and a "Max" button are used for entering the exact amount of BIFI you want to deposit. Furthermore, the Beefy Vault Fees ([# What is the vault fee structure?](#)) are shown.

### 6. An example deposit:

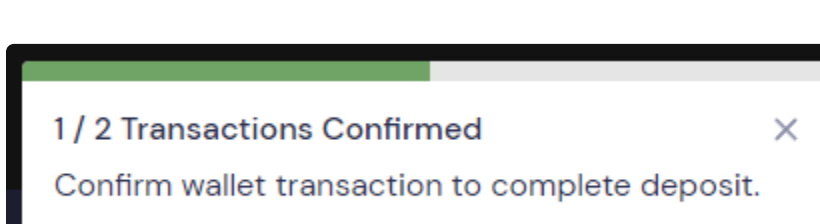
In this example, we first click the "Max" button to deposit all the BIFI in our wallet, followed by clicking on the "Deposit All" button.



When it is the first time you deposit in this vault, you need to grant permission to the vault's contract and allow it to access your funds:

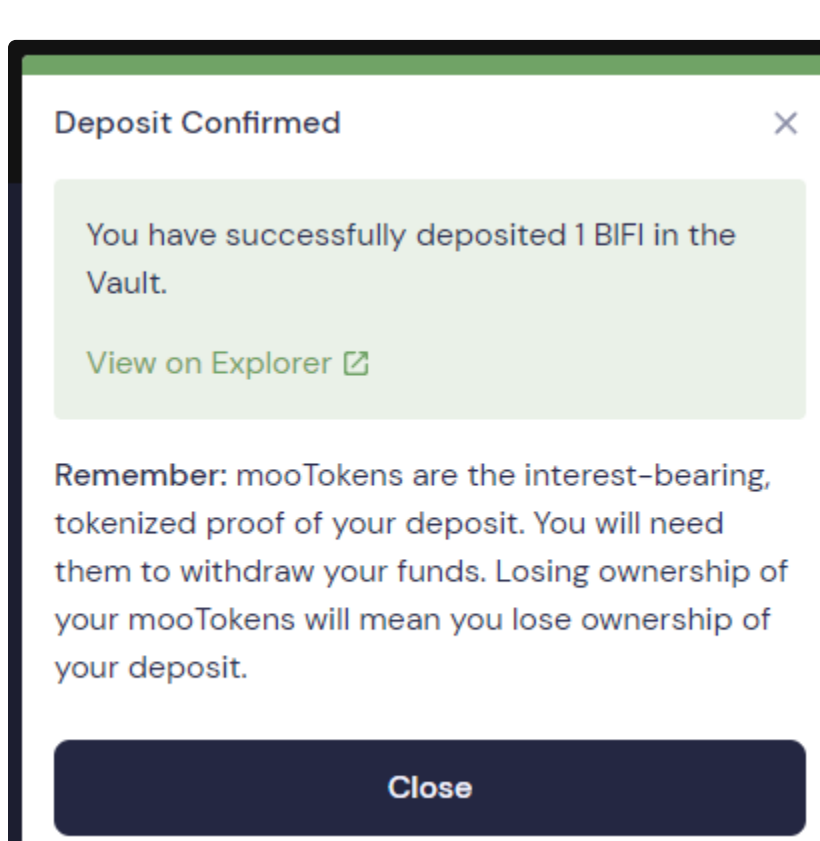


In the next transaction you will be actually depositing in the vault:



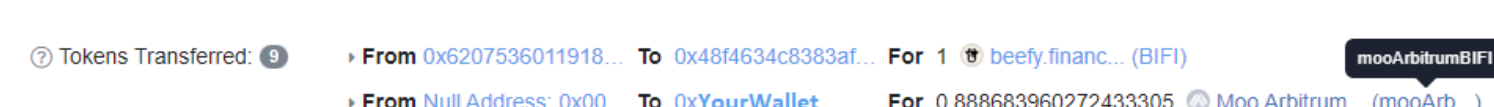
In summary, depositing into a new vault always requires two transactions: one for approving the spending permission and one for the actual deposit.

### 7. Deposit confirmation:



Once the transaction succeeds, a message will pop up confirming the deposit and it contains a link to the transaction in the block explorer. It is very important to understand that your wallet now holds a tokenized proof of deposit called a "mooToken" ([# What are mooTokens?](#)). This mooToken is required to withdraw from the vault, don't lose it!

On the block explorer page of the deposit transaction you can find out that the mooTokens are indeed supplied to your wallet after depositing in the vault. The token transfer will look something like this:



Since mooTokens are interest-bearing, they are more valuable than their "normal" token counterpart. This is also the reason why the mooToken amount does not 1:1 match with the token amount initially deposited ([# How do mooTokens earn interest?](#)).

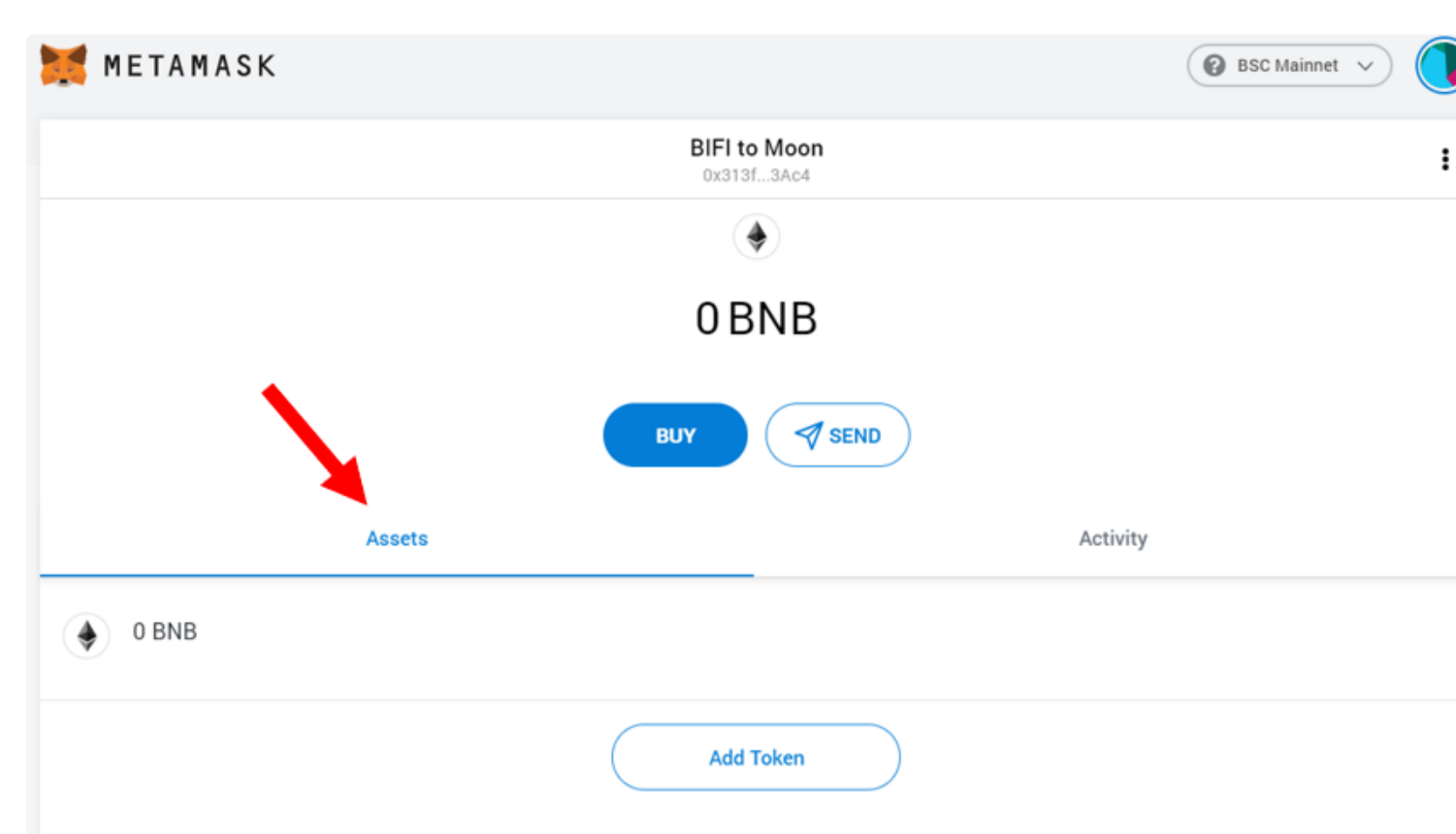
That's it, once the harvest function on this vault is called, you are already earning yield!

# How to Add a Custom Token to Metamask

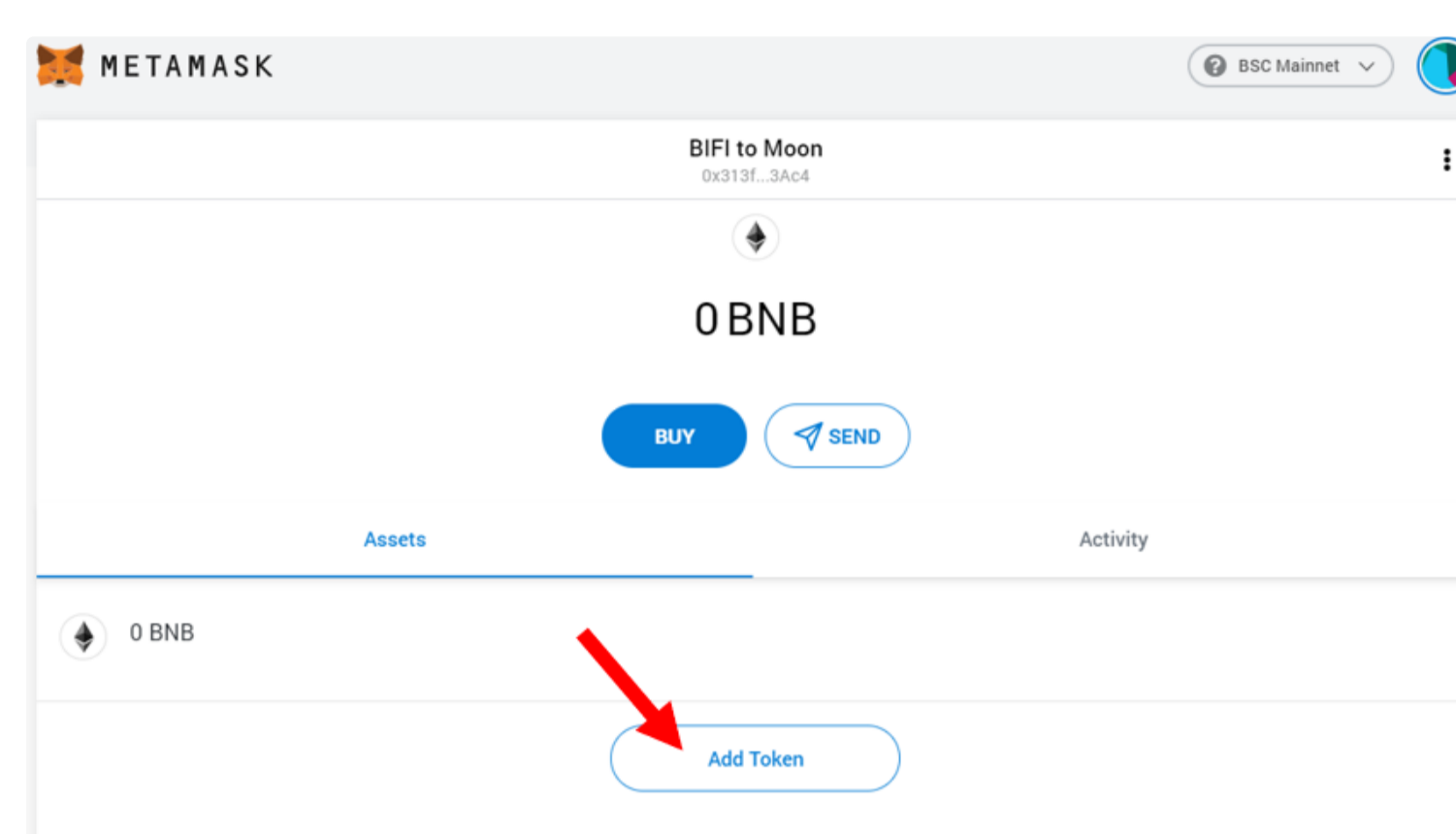
Here's how to add a custom token to Metamask. This guide uses BIFI as an example.

## Visual Walkthrough

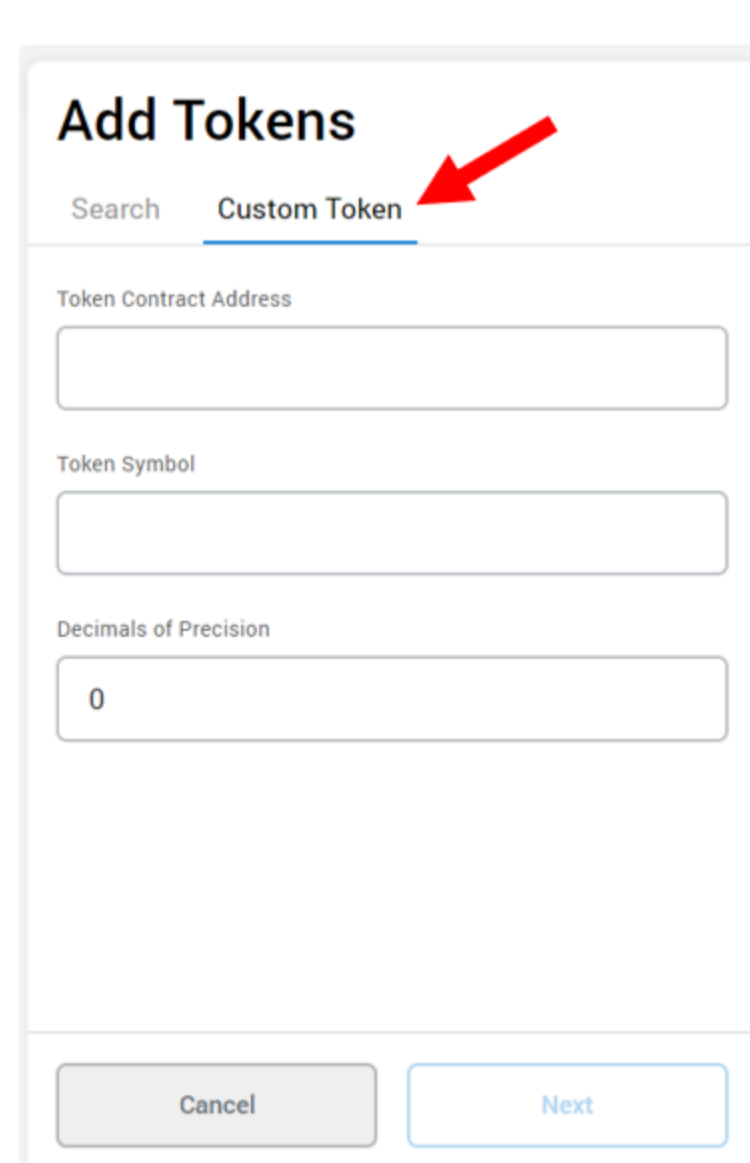
1. Open Metamask and click 'Assets' to see the tokens in your wallet



2. Scroll down to the bottom and click 'Add Token'.



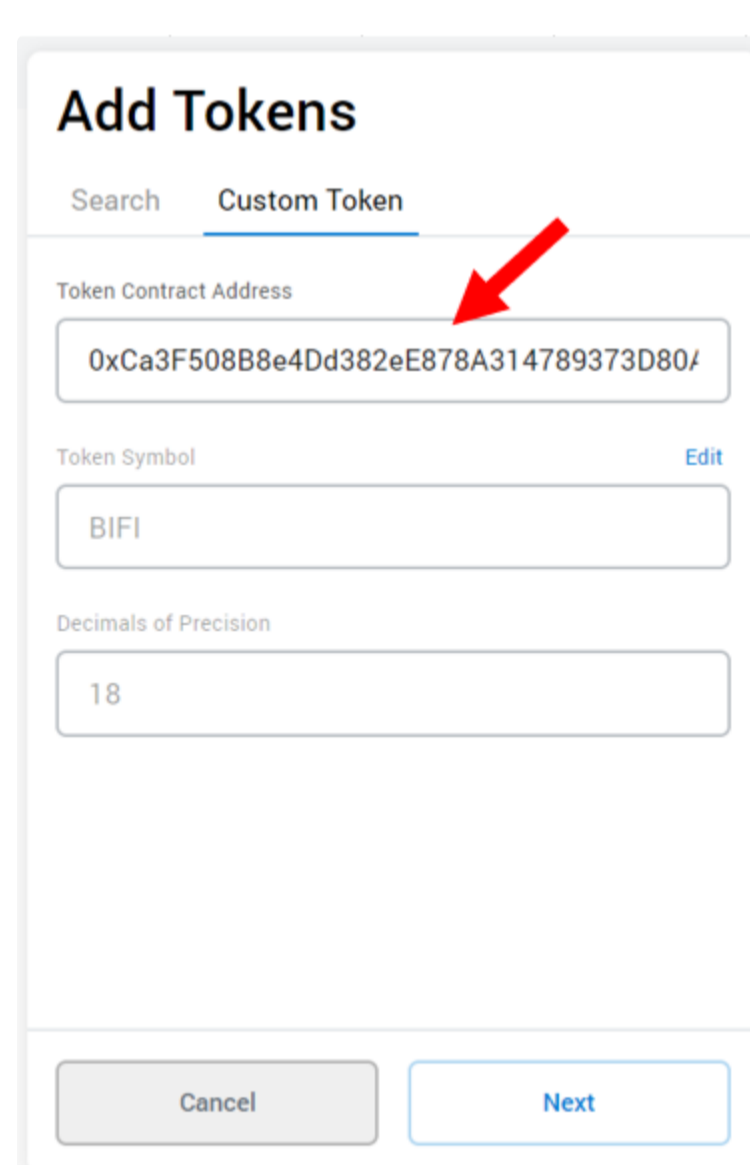
3. Click 'Custom Token'

A screenshot of the 'Add Tokens' dialog box in Metamask. It has two tabs: 'Search' and 'Custom Token'. The 'Custom Token' tab is selected and highlighted with a red arrow. Below the tabs are three input fields: 'Token Contract Address', 'Token Symbol', and 'Decimals of Precision'. At the bottom are 'Cancel' and 'Next' buttons.

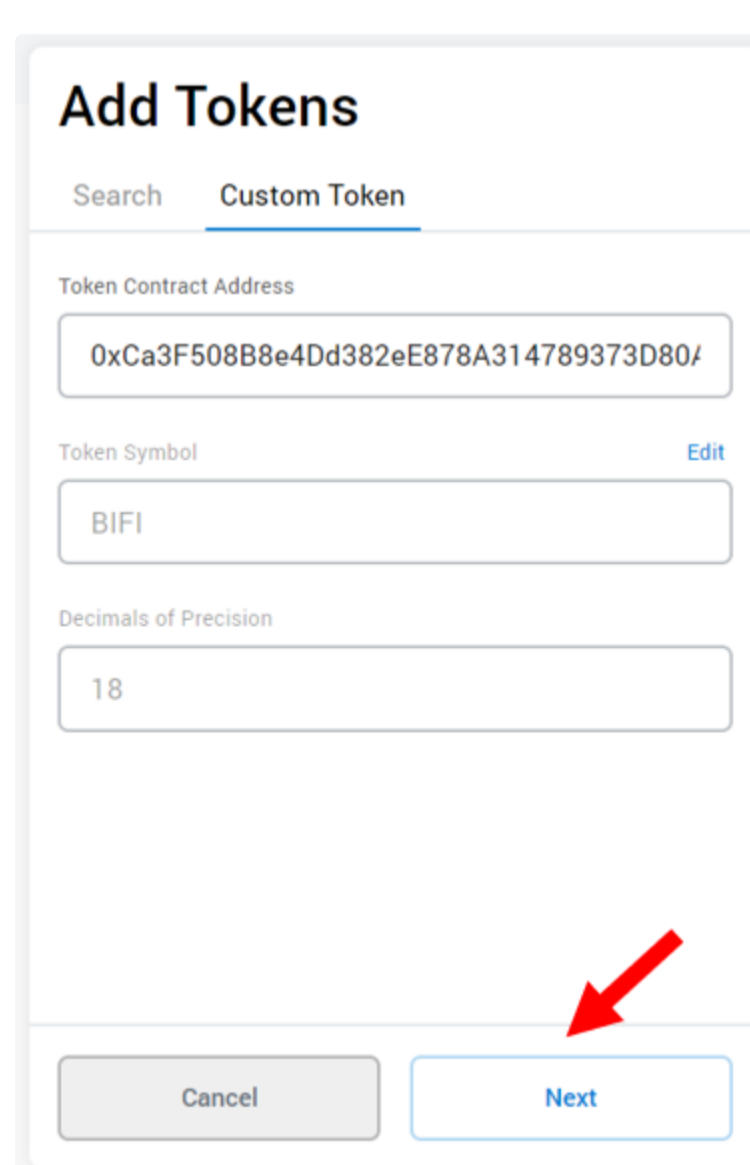
4. Past the contract address for BIFI into the 'Token Contract Address' field

BIFI token BNB Chain Contract Address : 0xCa3F508B8e4Dd382eE878A314789373D80A

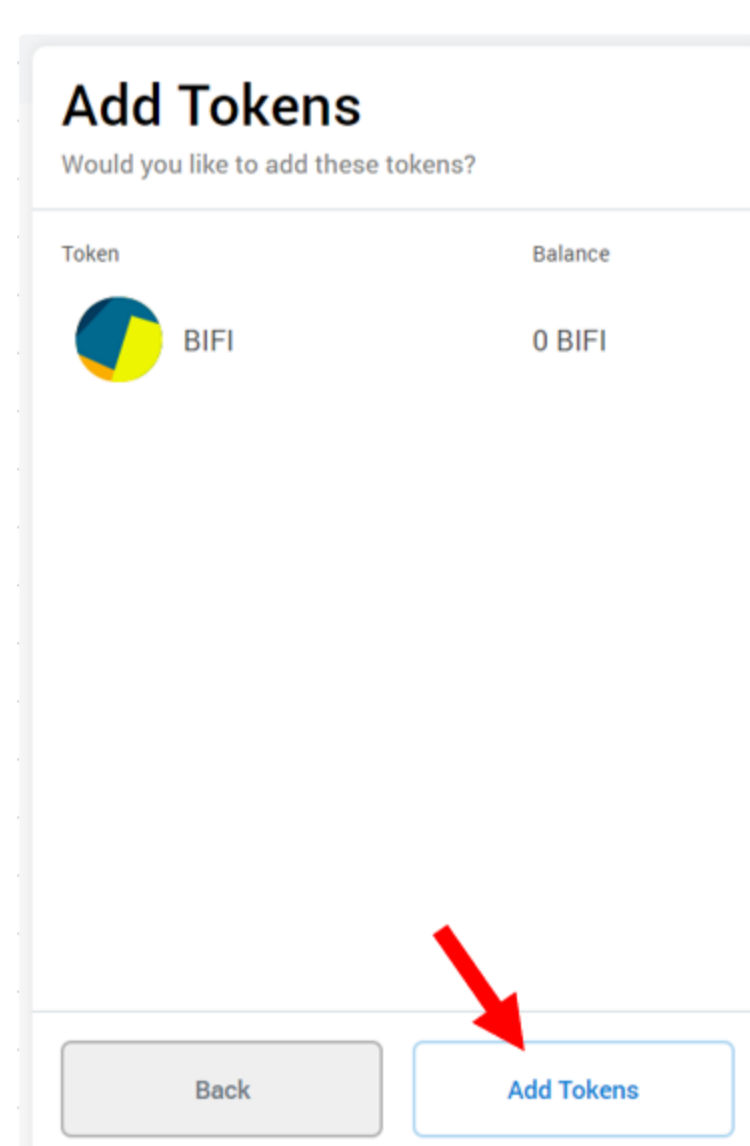
The BIFI token Contract Address on other chains can be found here: [Contract Addresses](#)

A screenshot of the 'Add Tokens' dialog box. The 'Token Contract Address' field now contains the BIFI contract address: '0xCa3F508B8e4Dd382eE878A314789373D80A'. A red arrow points to this field. The 'Token Symbol' field contains 'BIFI' and the 'Decimals of Precision' field contains '18'. 'Cancel' and 'Next' buttons are at the bottom.

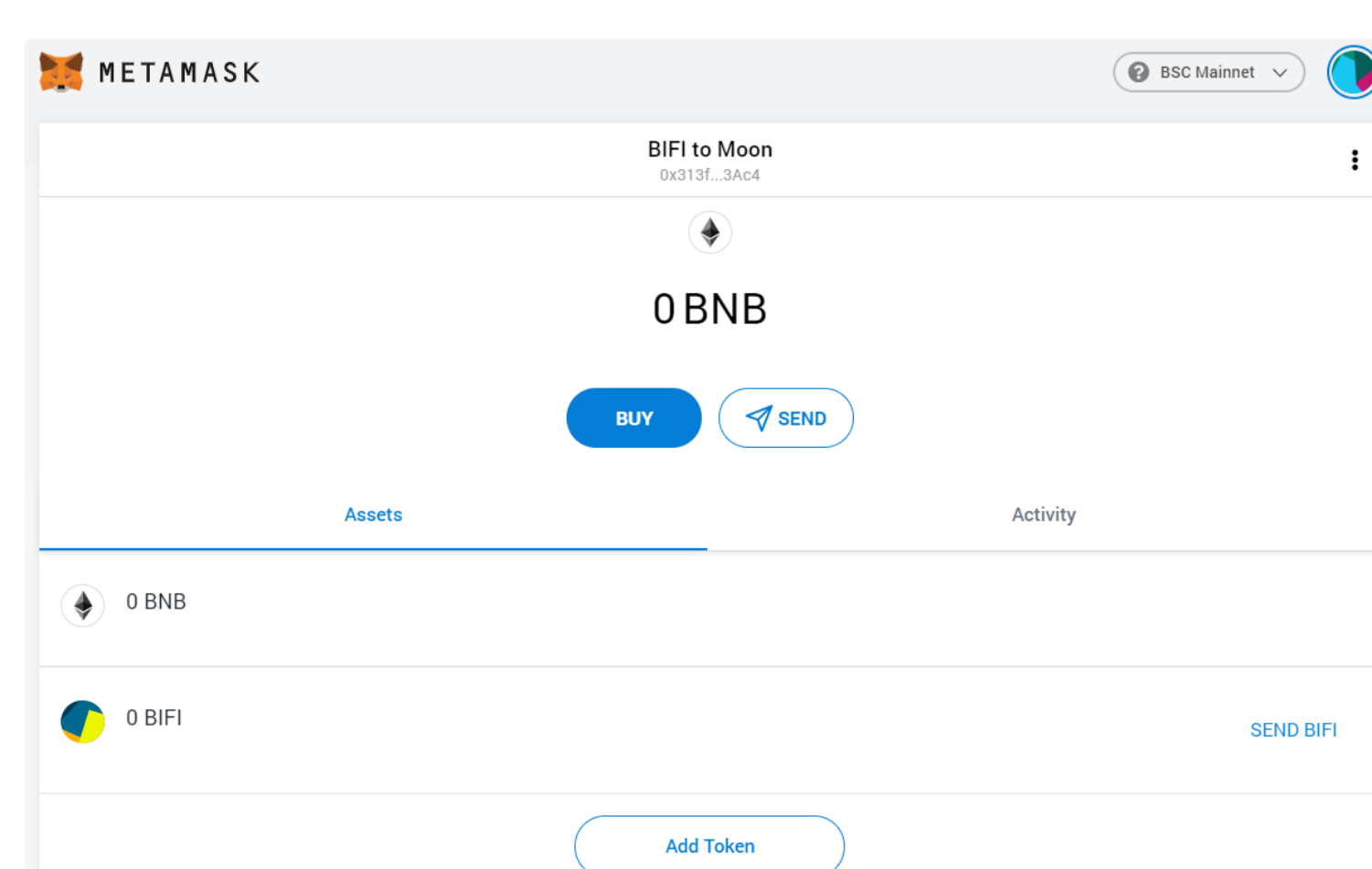
5. Click 'Next'

A screenshot of the 'Add Tokens' dialog box, showing the same information as the previous step. A red arrow points to the 'Next' button at the bottom right.

6. Click 'Add Tokens' to add the new token

A screenshot of the 'Add Tokens' dialog box showing a confirmation screen. It asks 'Would you like to add these tokens?' and displays a table with one row: 'Token' (BIFI icon) and 'Balance' (0 BIFI). At the bottom are 'Back' and 'Add Tokens' buttons. A red arrow points to the 'Add Tokens' button.

7. Mooooo! BIFI should appear in your assets list so it's easier to track and use

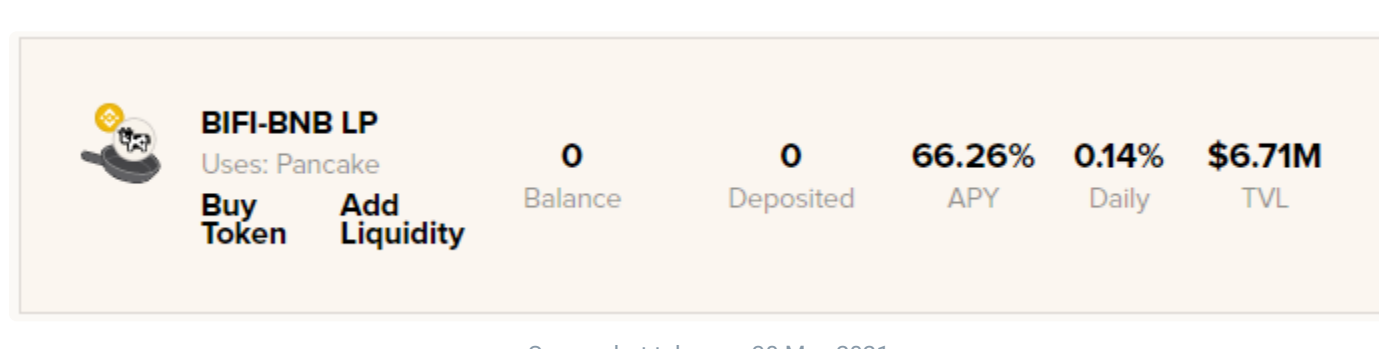




# How to Add and Remove Liquidity

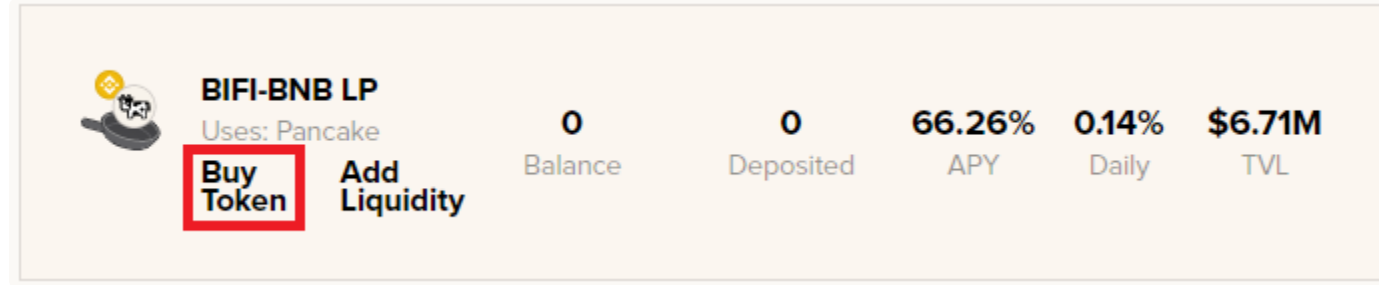
In this guide you will find the required steps how to Add and Remove Liquidity as well as staking and unstaking it from the vault.

As an example, we are going to work with BIFI-BNB LP in this guide. In a liquidity pool, both BIFI and BNB need to be provided at a 50/50 ratio value wise. Since we start with 100% BNB, this guide covers swapping BNB to BIFI too.



## Adding liquidity

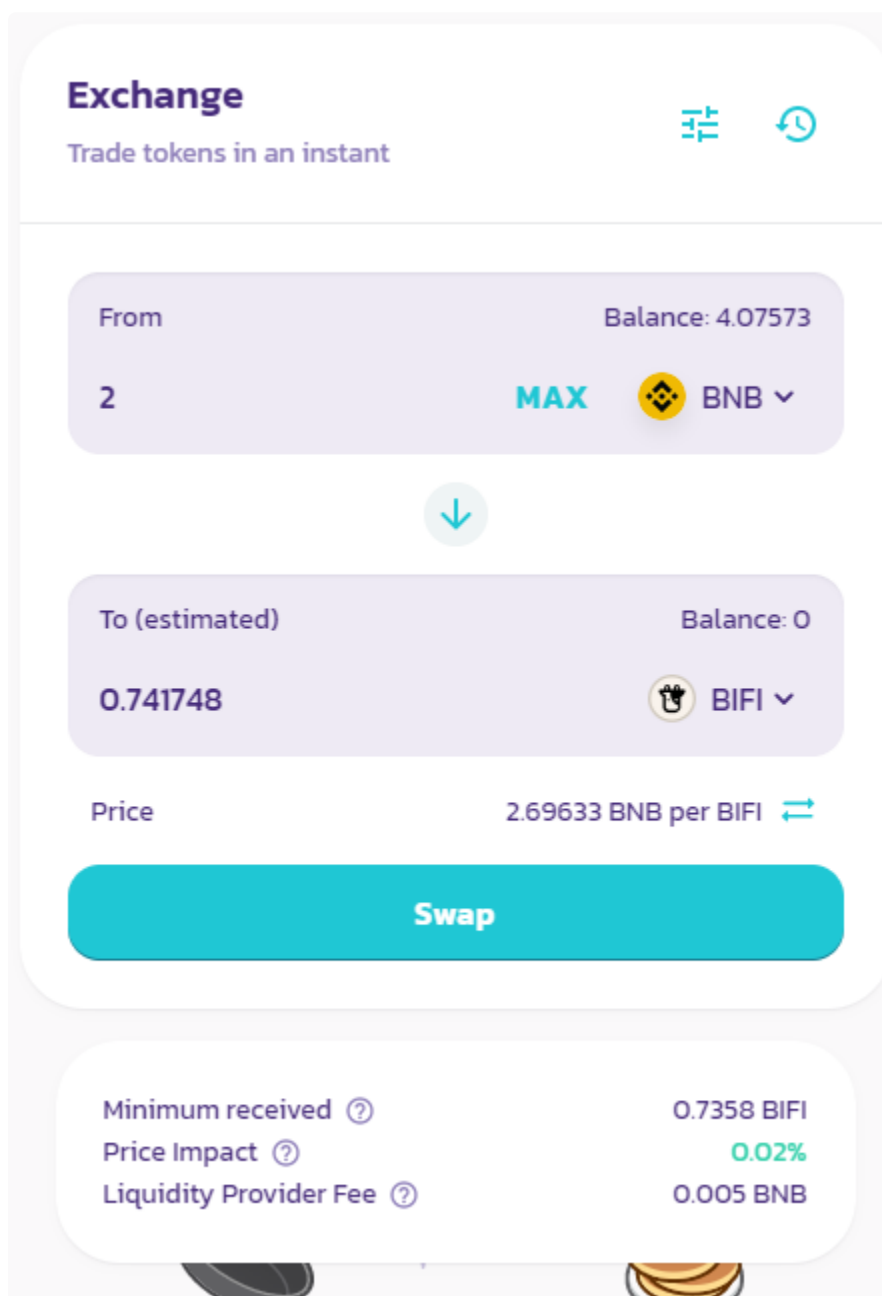
### 1. Click "Buy Token"



### 2. Confirm the "Token imported" screen on PancakeSwap

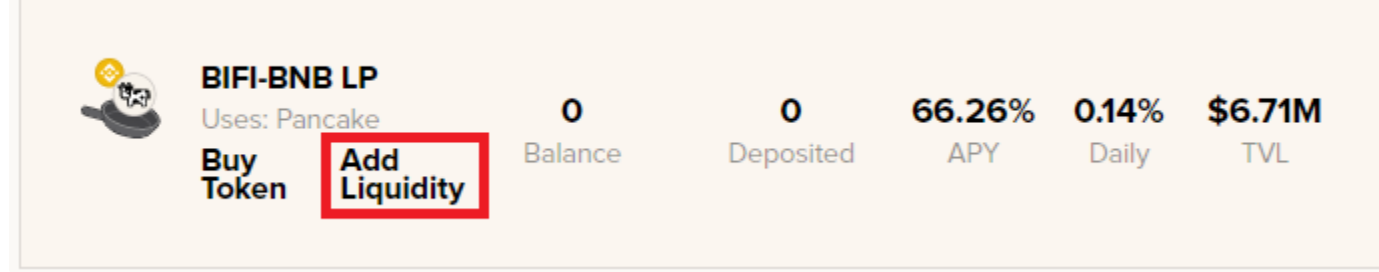
### 3. Swap BNB for BIFI

Our wallet currently holds 4.0757 BNB. We will use a maximum total of 4 BNB to provide liquidity. Since we need to provide liquidity at a 50/50 ratio value wise, we will need to swap 2 BNB for BIFI first.

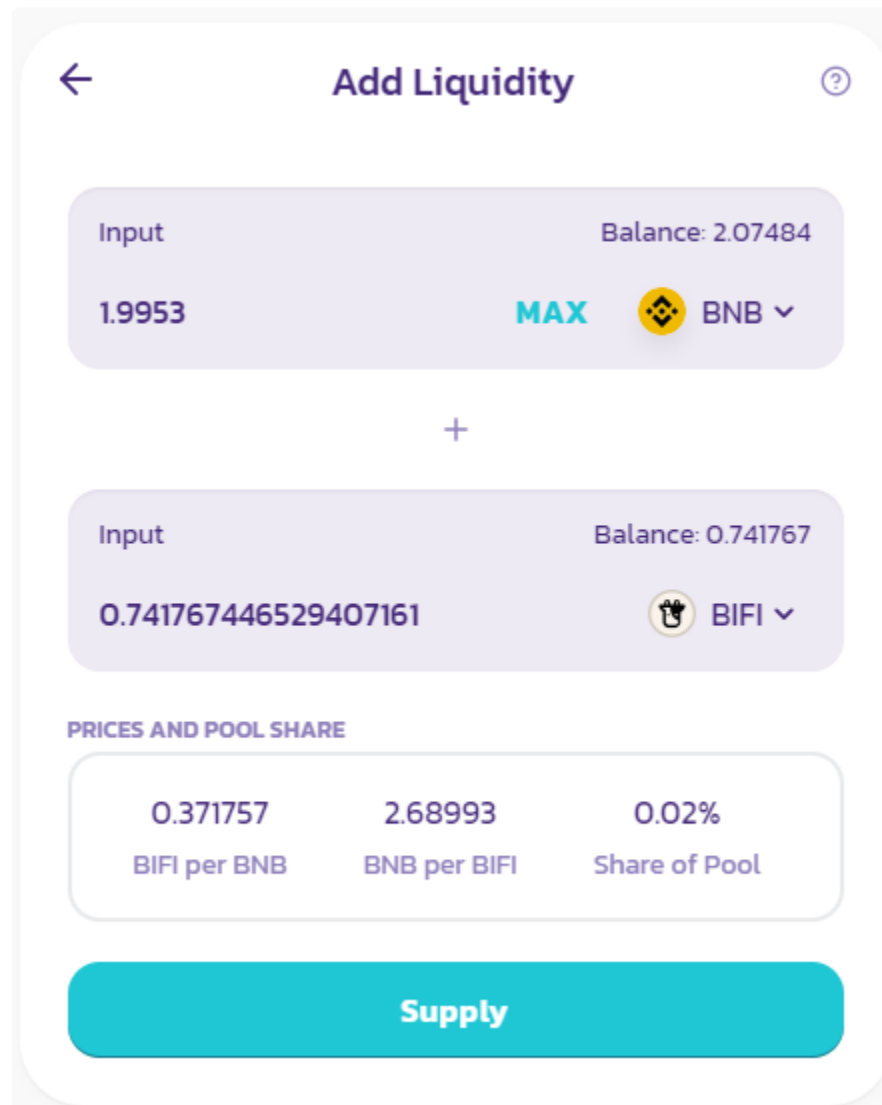


Confirm Swap in the next pop-up screen.

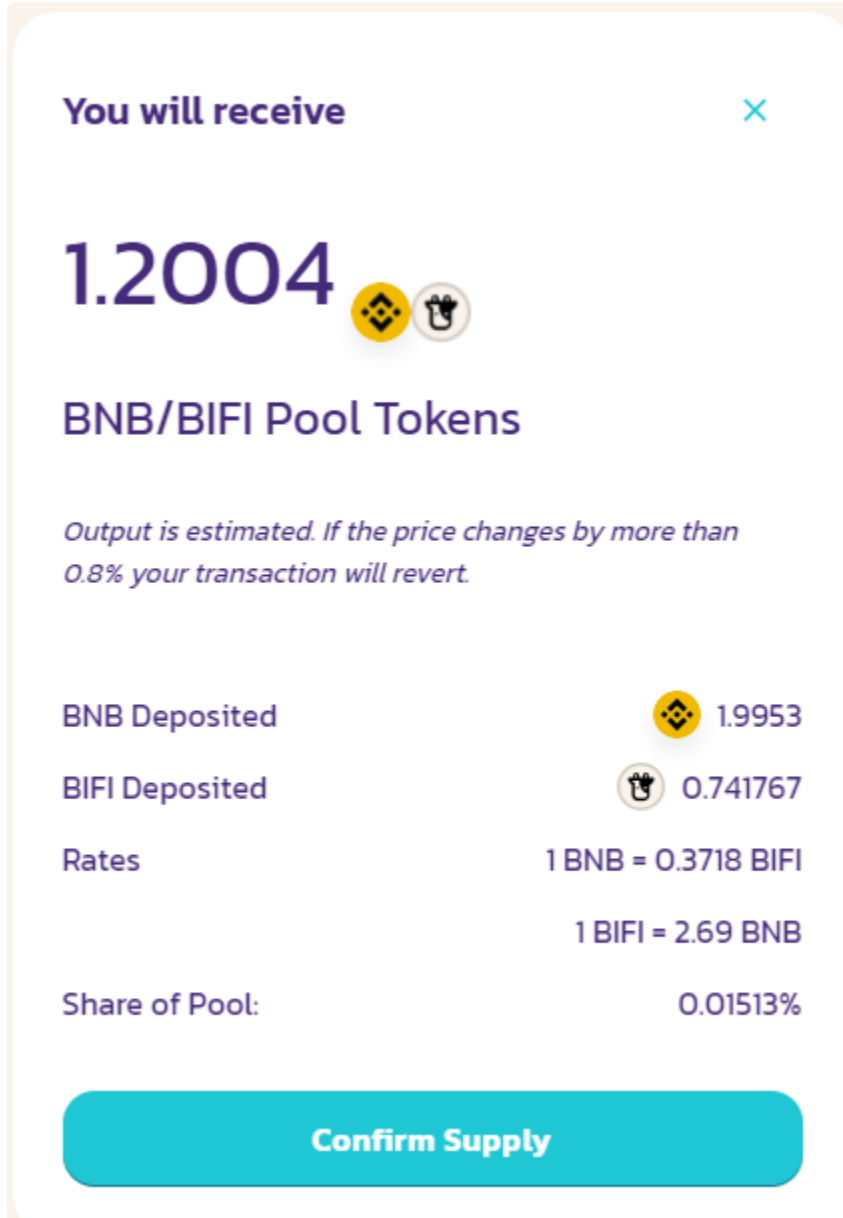
### 4. Click "Add Liquidity"



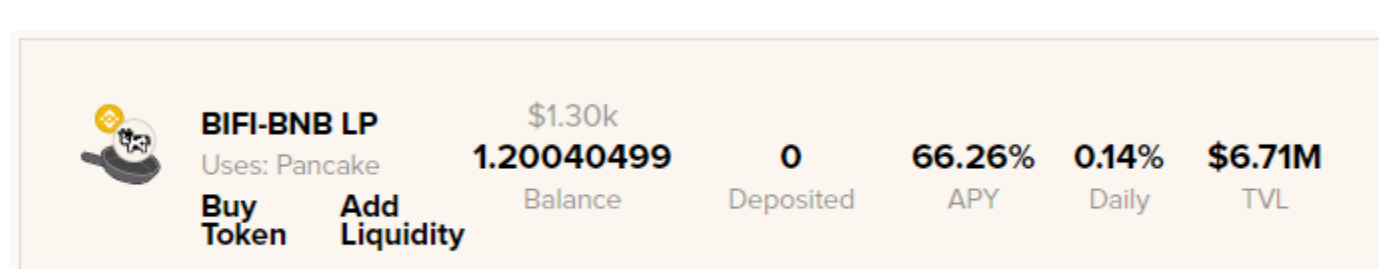
### 5. Click "MAX" for BIFI input, Approve BIFI and Supply Liquidity



### 6. Confirm Supply

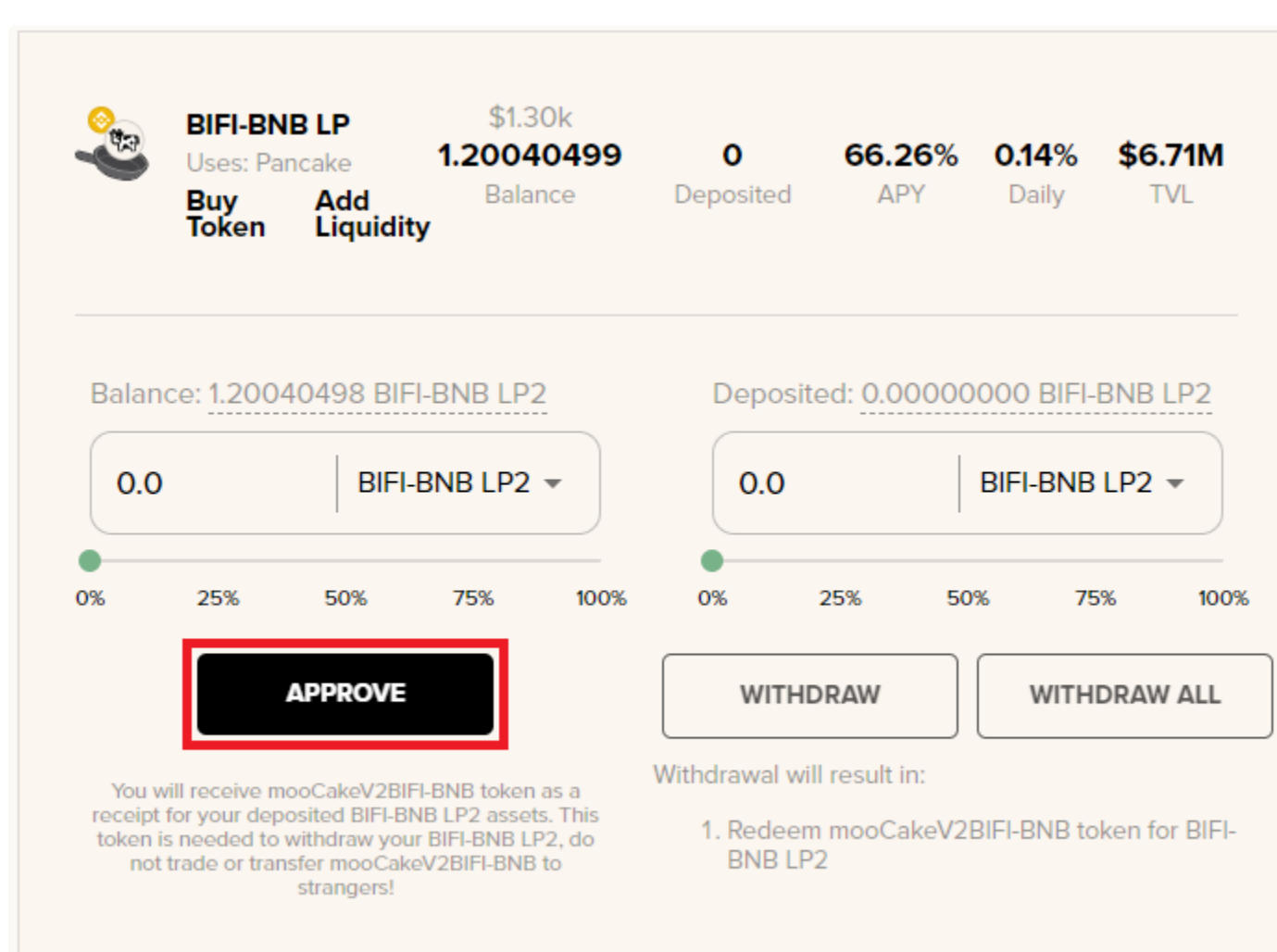


### 7. The vault now shows a balance!

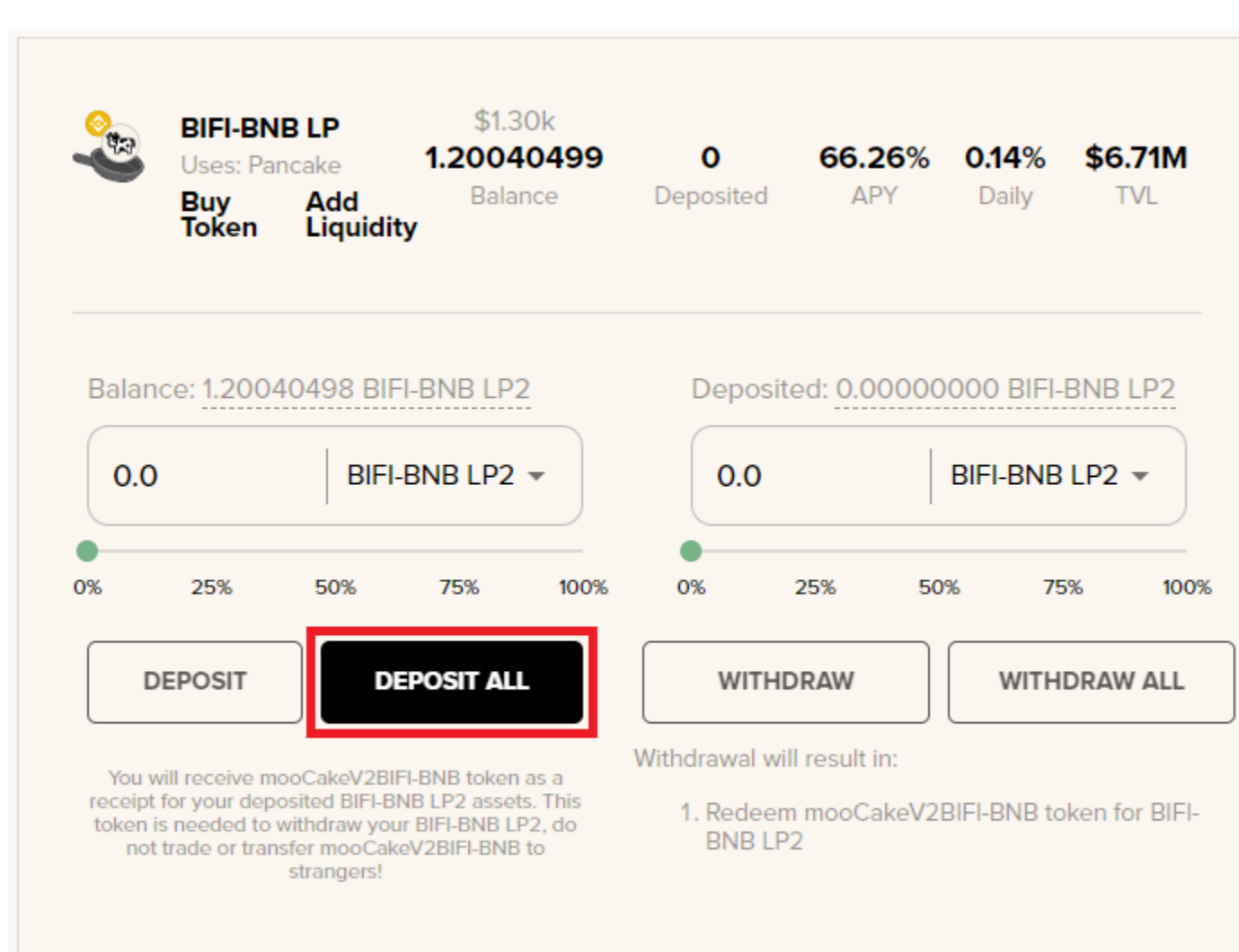


Click on the vault to open up the deposit and withdraw menu.

### 8. "Approve"



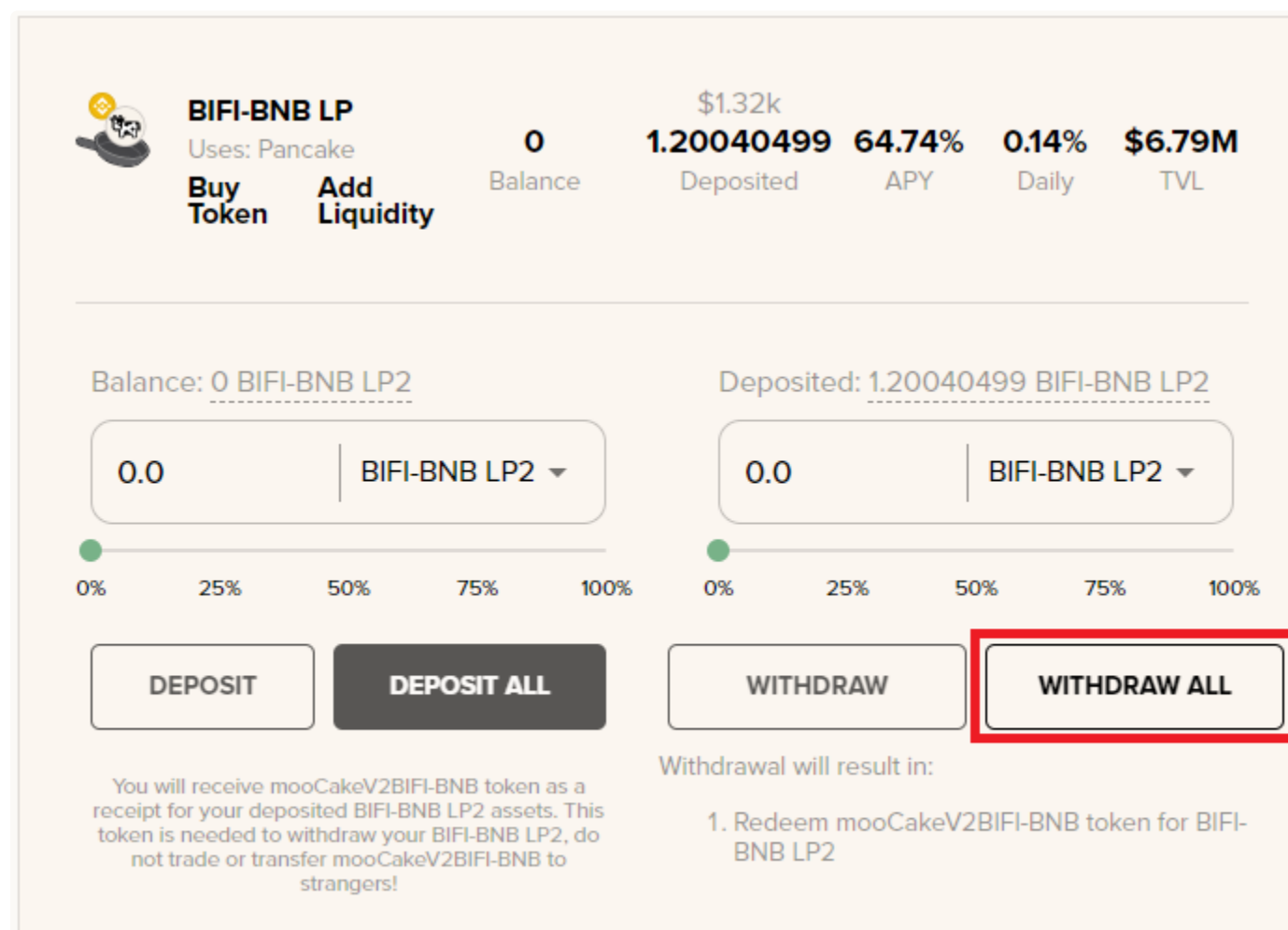
### 9. "Deposit All"



That's it! We now created liquidity and deposited BIFI-BNB LP in the vault. You can check this guide to see when the vault will harvest rewards and compound for more BIFI-BNB LP tokens.

## Removing Liquidity

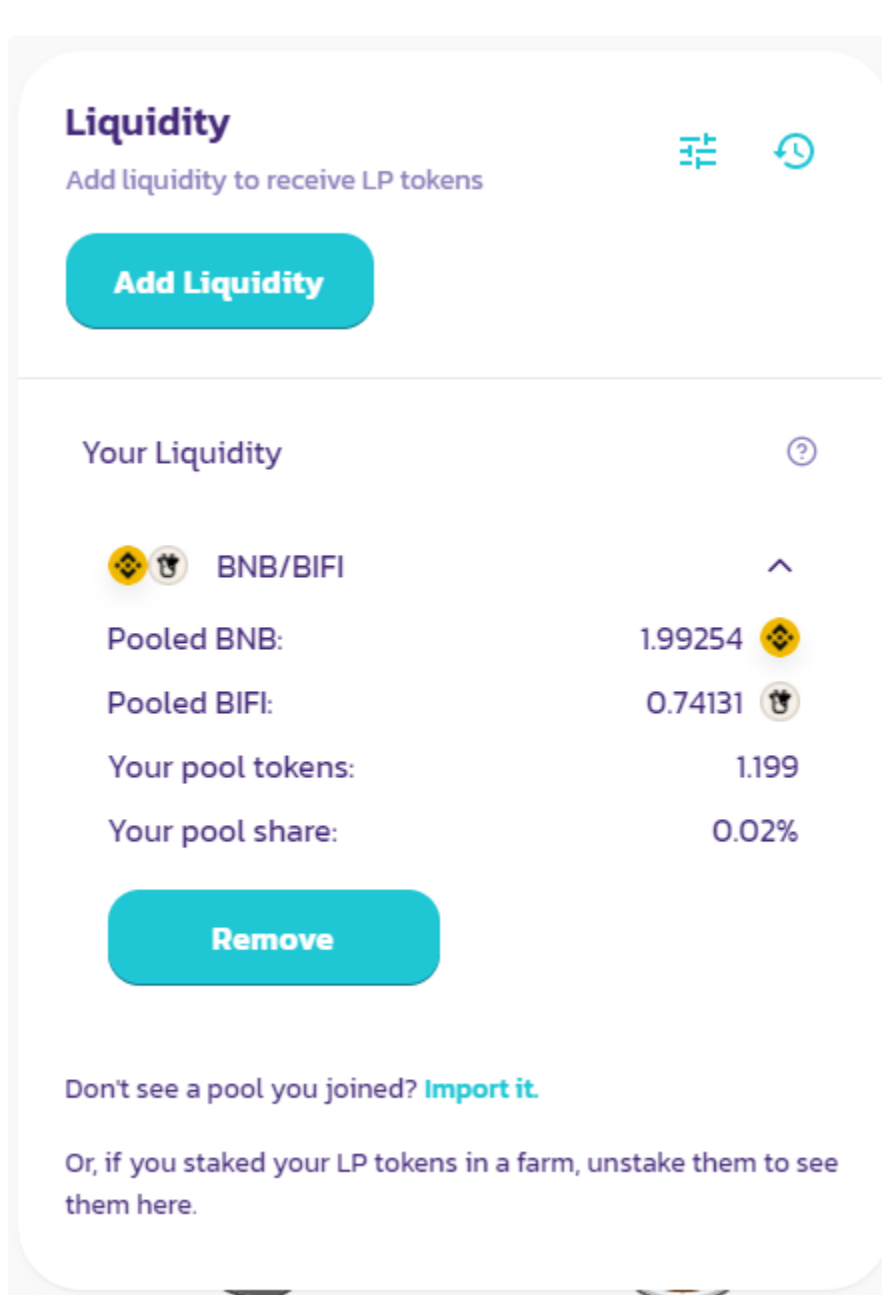
### 1. "Withdraw All"



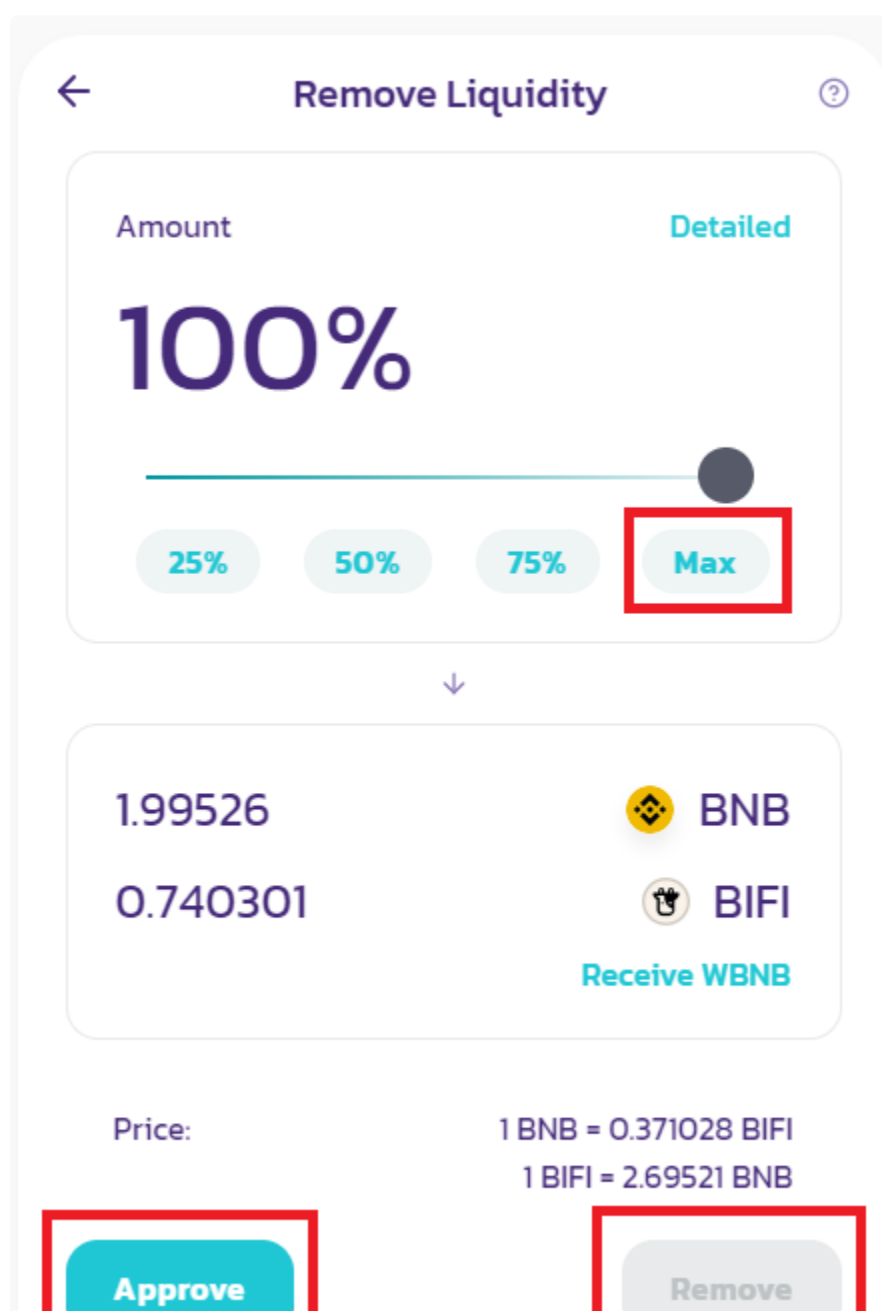
Note: withdrawal fees will be deducted from your deposited token amount.

### 2. Go back to PancakeSwap

and head over to the Liquidity section. It will show the BIFI-BNB LP tokens under "Your Liquidity"

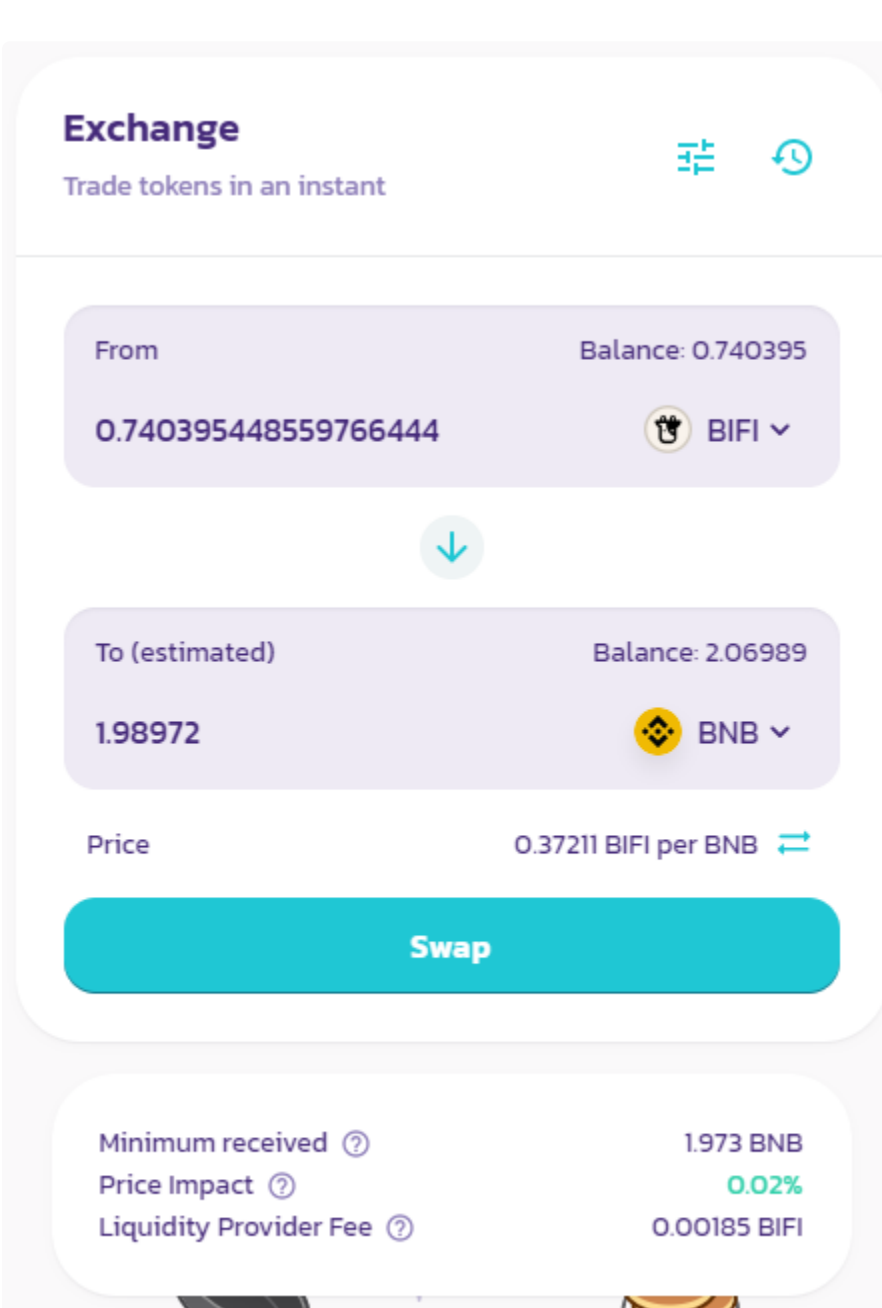


### 3. Click "Remove"



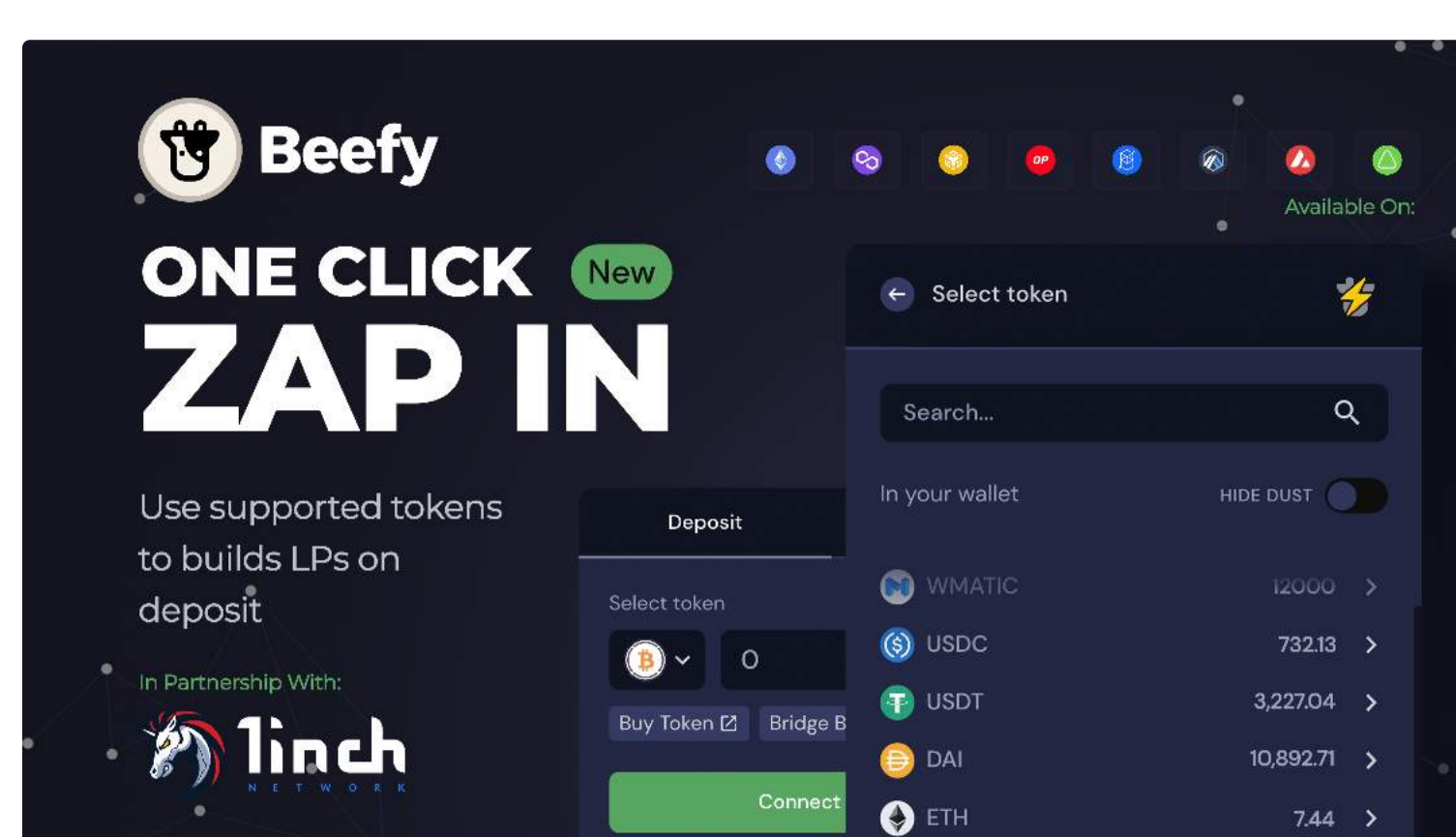
In the next screen, click "Max" and Approve", and then "Remove".

### 4. Optional: Swap BIFI back to BNB



# How to use Beefy ZAP

One-click Beefy Vault Investing!



Beefy ZAP lets you create liquidity pool tokens and deposit into Beefy vaults with just one transaction. You no longer need to manually add and remove liquidity! Our ZAP tooling is a simple, quick and safe solution that eliminates the need for you to handle complicated LP tokens or obscure tokens, and even saves you from needing to leave the comfy environs of the Beefy app.

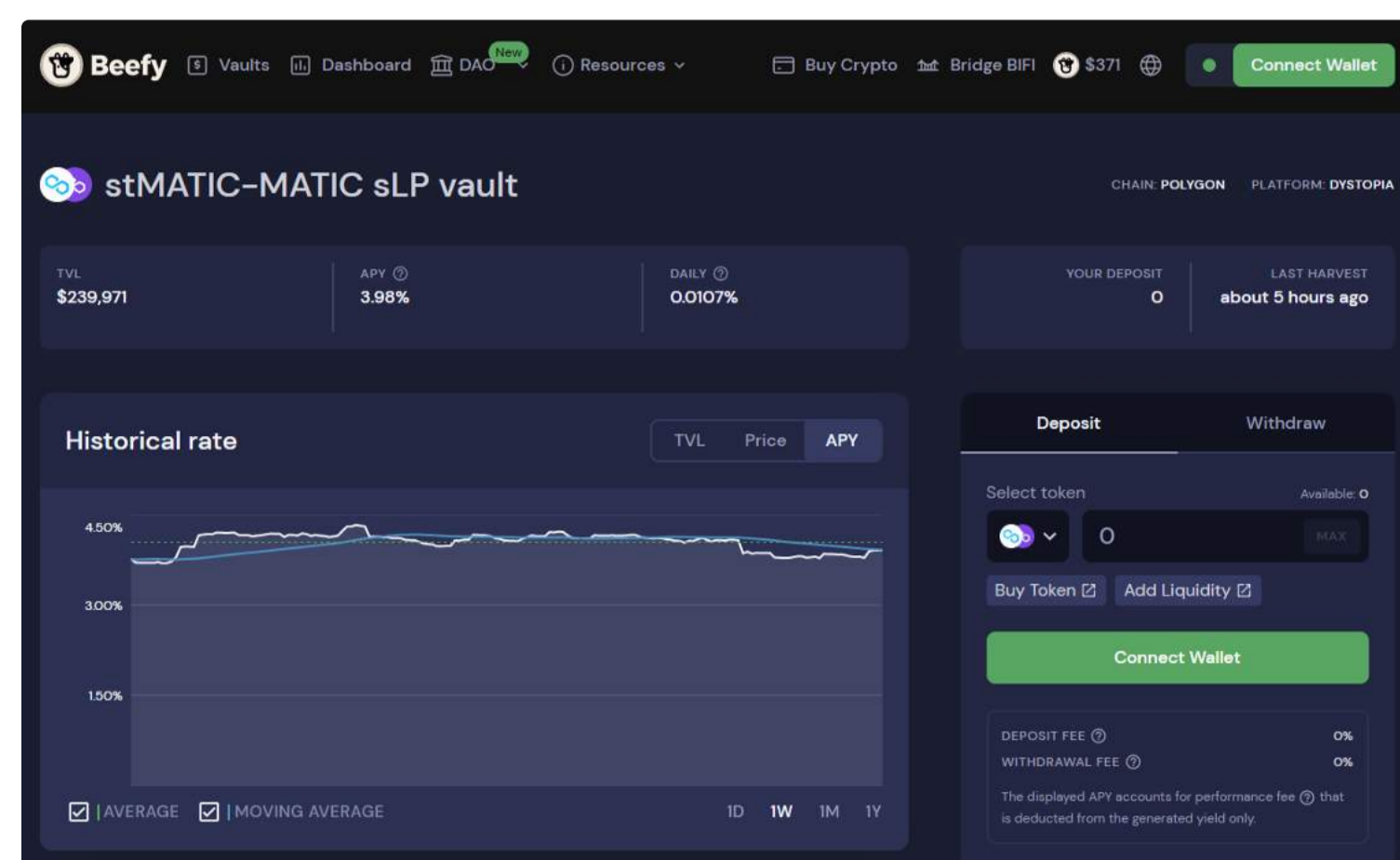
This page sets out a short tutorial on depositing and withdrawing into Beefy vaults with our different ZAP tools. For a higher-level conceptual explanation of our ZAP tooling, see the [#Beefy ZAP](#) infographics section.

When using ZAP always check your quote! While ZAP does protect you against market slippage (price changes between the time of order and time of fulfillment), it does not protect you against price impact (how much your transaction will change the price of the tokens in the liquidity pool). As a general rule, the smaller the liquidity of the asset, the larger the risk of price impact.

## Example Vault

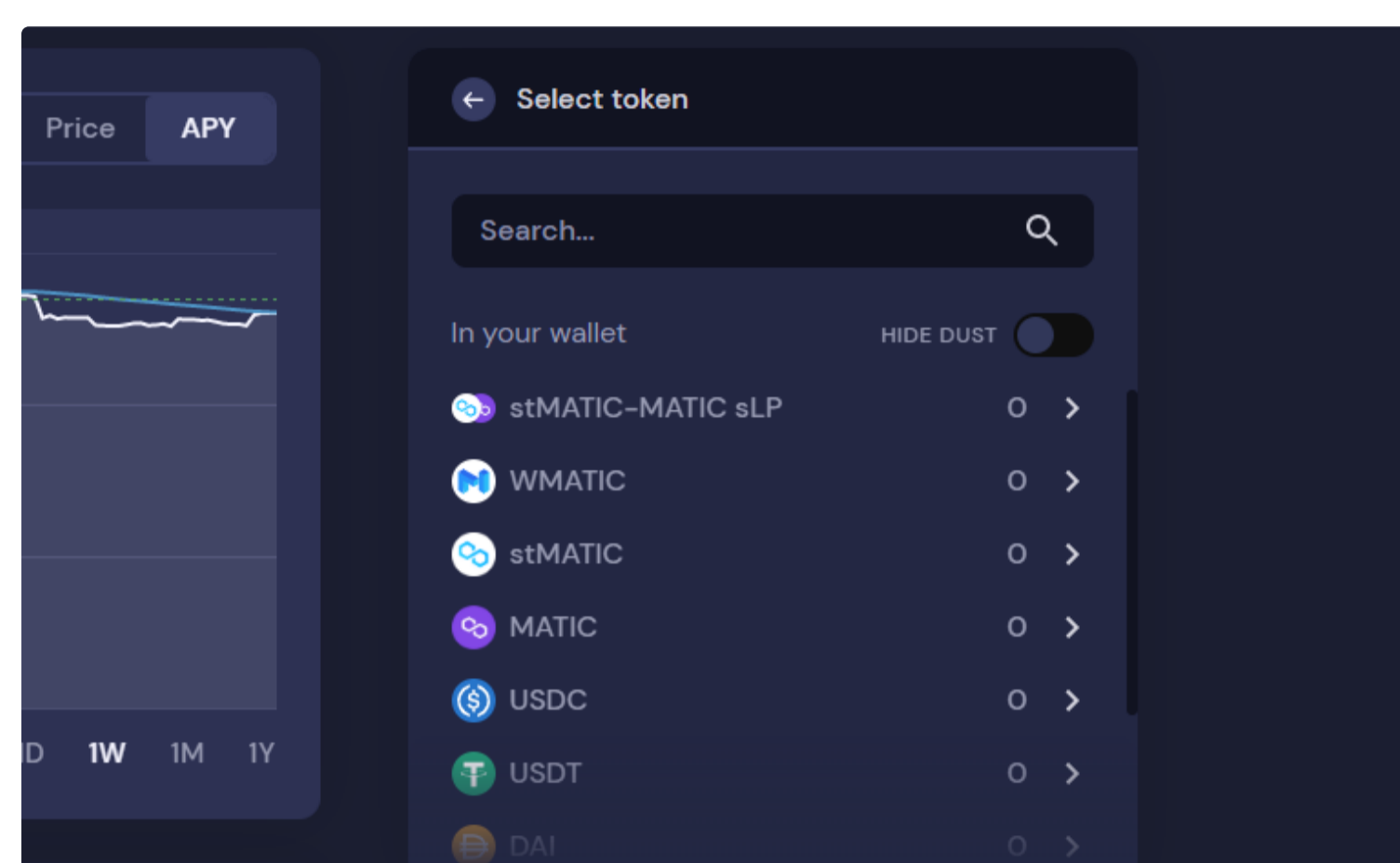
Last Update: January 2023. As the site is under constant development, the interface that you see may vary slightly from the below screenshots over time.

For the purposes of these tutorials, we will use Beefy's stMATIC-MATIC sLP vault for the Dystopia DEX on the Polygon blockchain. This vault supports both ZAP V1 and ZAP V2.



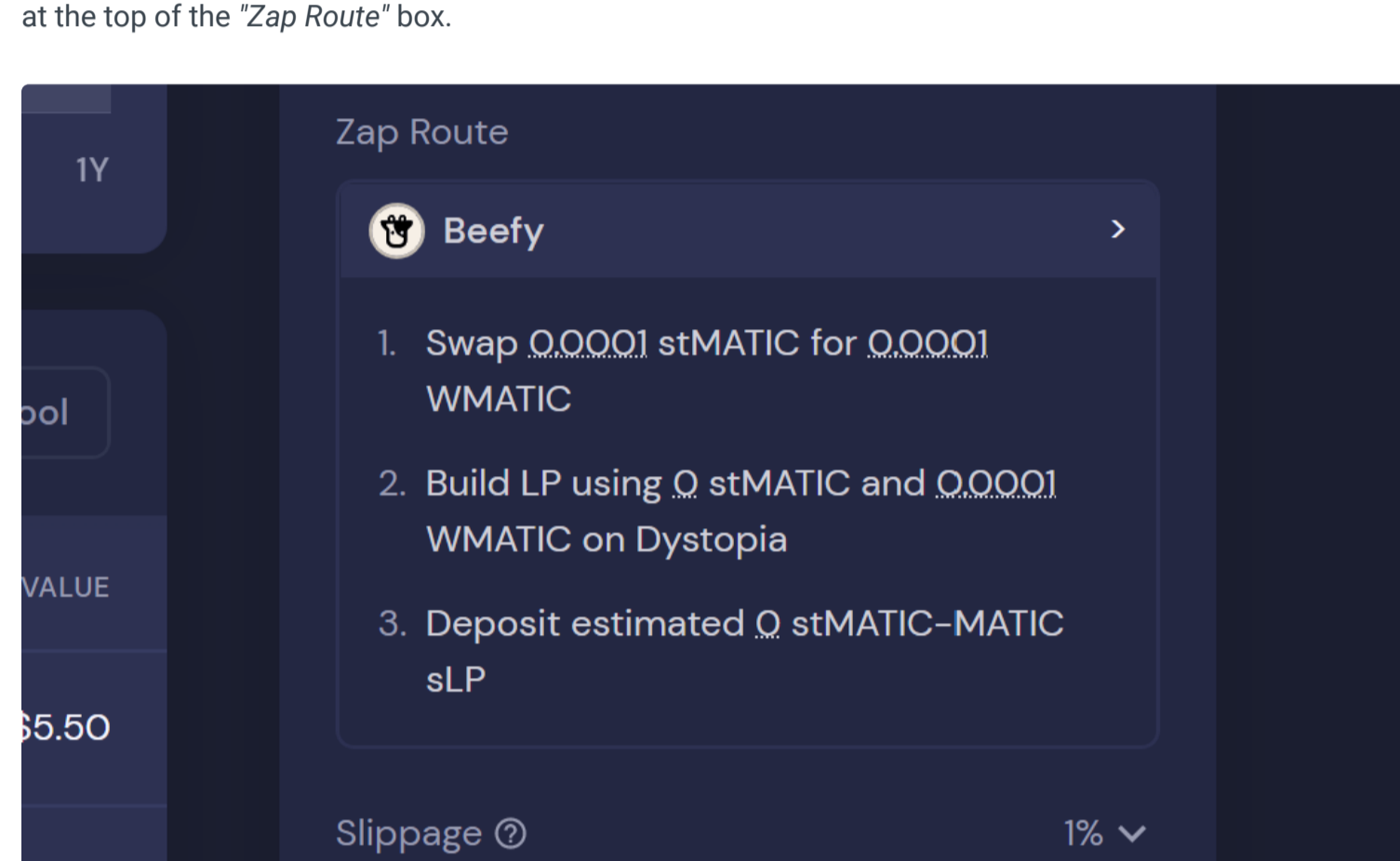
The first step is to navigate to the relevant page for the vault you want to enter. For this tutorial we will use the stMATIC-MATIC sLP vault for the Dystopia DEX on the Polygon blockchain.

For those following along with this tutorial, just navigate to the relevant page for the vault you would like to enter. However, be aware that our ZAP tools are not available for every vault on every blockchain, so you should check at this stage whether the deposit token dropdown menu under 'Select token' does permit deposits with different assets. Make sure to also 'Connect Wallet' before trying to deposit.



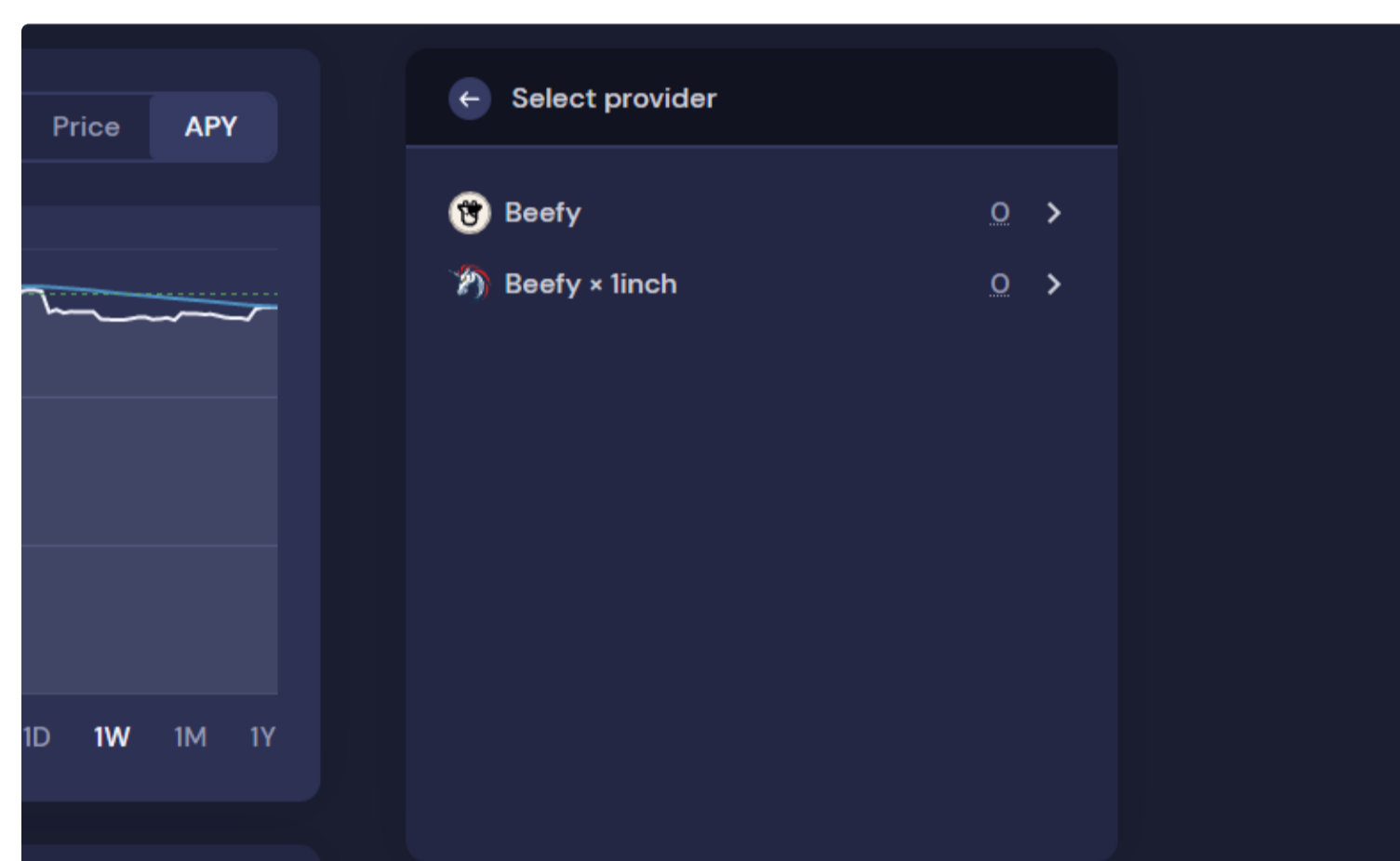
The second step is to check which tokens you can use for the deposit. Without ZAP you can only deposit the vault asset (e.g. the LP token at the top of the list). For V1, you can deposit the underlying assets of the vault (e.g. WMATIC, stMATIC or MATIC). For V2, you can deposit from a broader range of other assets (e.g. USDC, USDT, DAI, etc).

Where ZAP V2 is available and you select a token that can be used for either V1 or V2, you will also be able to decide which tool to use. To switch between the two, look for the side arrow on the 'Beefy' header at the top of the 'Zap Route' box.



Where both ZAP V1 and ZAP V2 are available for your chosen deposit asset, the Zap Route box allows you to select your preferred tool/provider to use. Click the side arrow on the 'Beefy' header.

The interface will then offer you to 'Select provider' and show all available ZAP providers and tools that you can use. In this case, you can select between ZAP V1 (or 'Beefy' shown below) or ZAP V2 (or 'Beefy x Tinch' shown below).

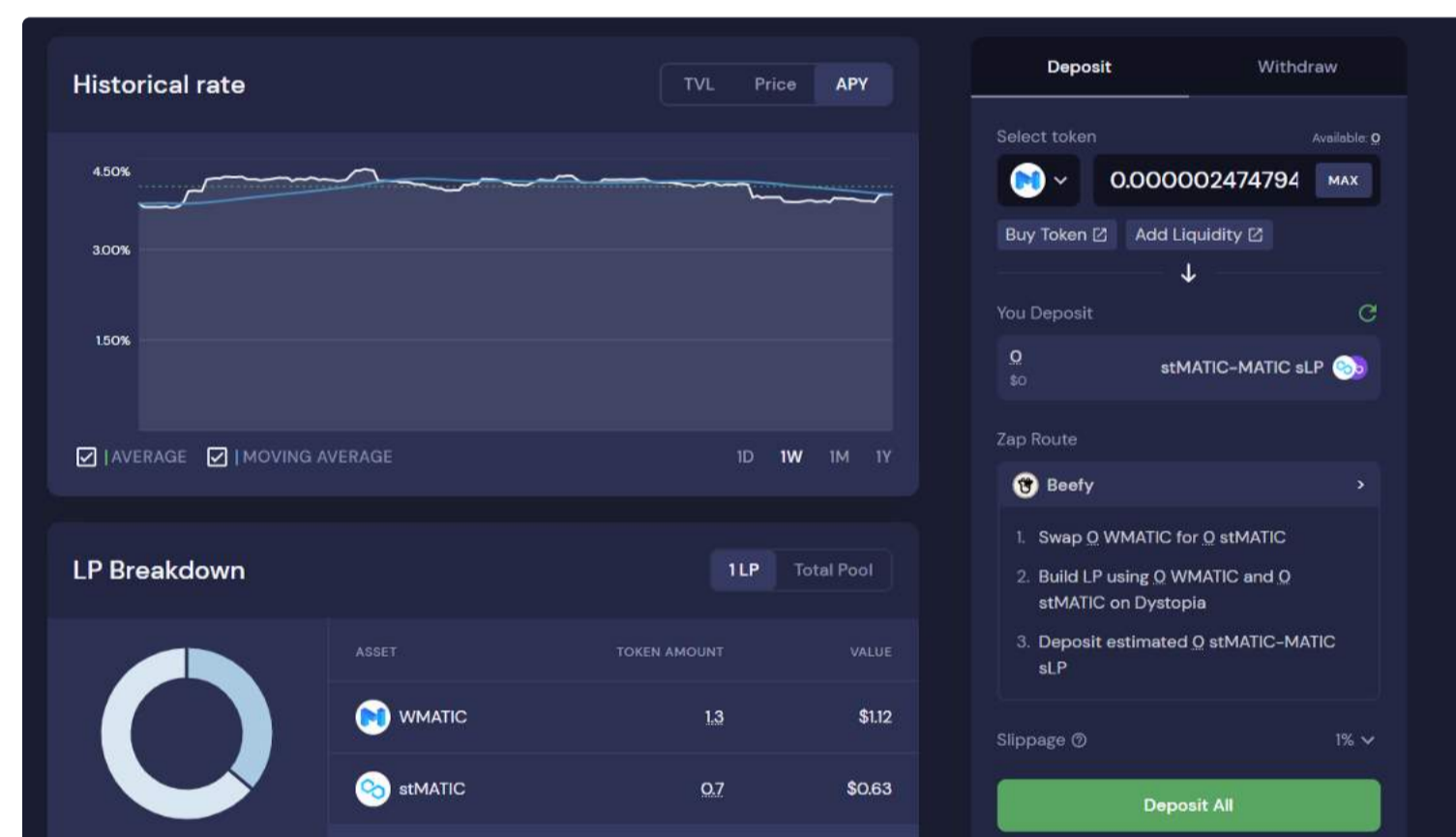


The interface allows you to select between using Beefy's own tool (using ZAP V1) or alternatively a swap aggregator like Tinch (using ZAP V2).

## ZAP V1: Entering Vaults

To use ZAP V1 for deposits, ensure you've selected the 'Deposit' tab at the top of the box shown at the right hand side in the image below. Use the 'Select token' dropdown to select one of the underlying assets for the vault (e.g. WMATIC, stMATIC or MATIC), but not the main deposit token (e.g. stMATIC-MATIC sLP). You can then enter the amount of the token you wish to deposit from your wallet, or hit 'MAX' to deposit all of your tokens at once.

The UI will then display an estimate for how much of the main deposit token (e.g. stMATIC-MATIC sLP) you will receive and then deposit through the ZAP V1 workflow. Below, the 'Zap Route' box shows the different steps of the workflow and estimates for how much of each token will be received and used in each part of the transaction. Please note that it is impossible to know in advance exactly how much you will receive for the swaps in a ZAP workflow, so the final amounts may vary slightly from the estimates.



For the ZAP V1 deposit workflow, select one of the underlying assets in the vault and input the quantity of that token you wish to deposit into the vault.

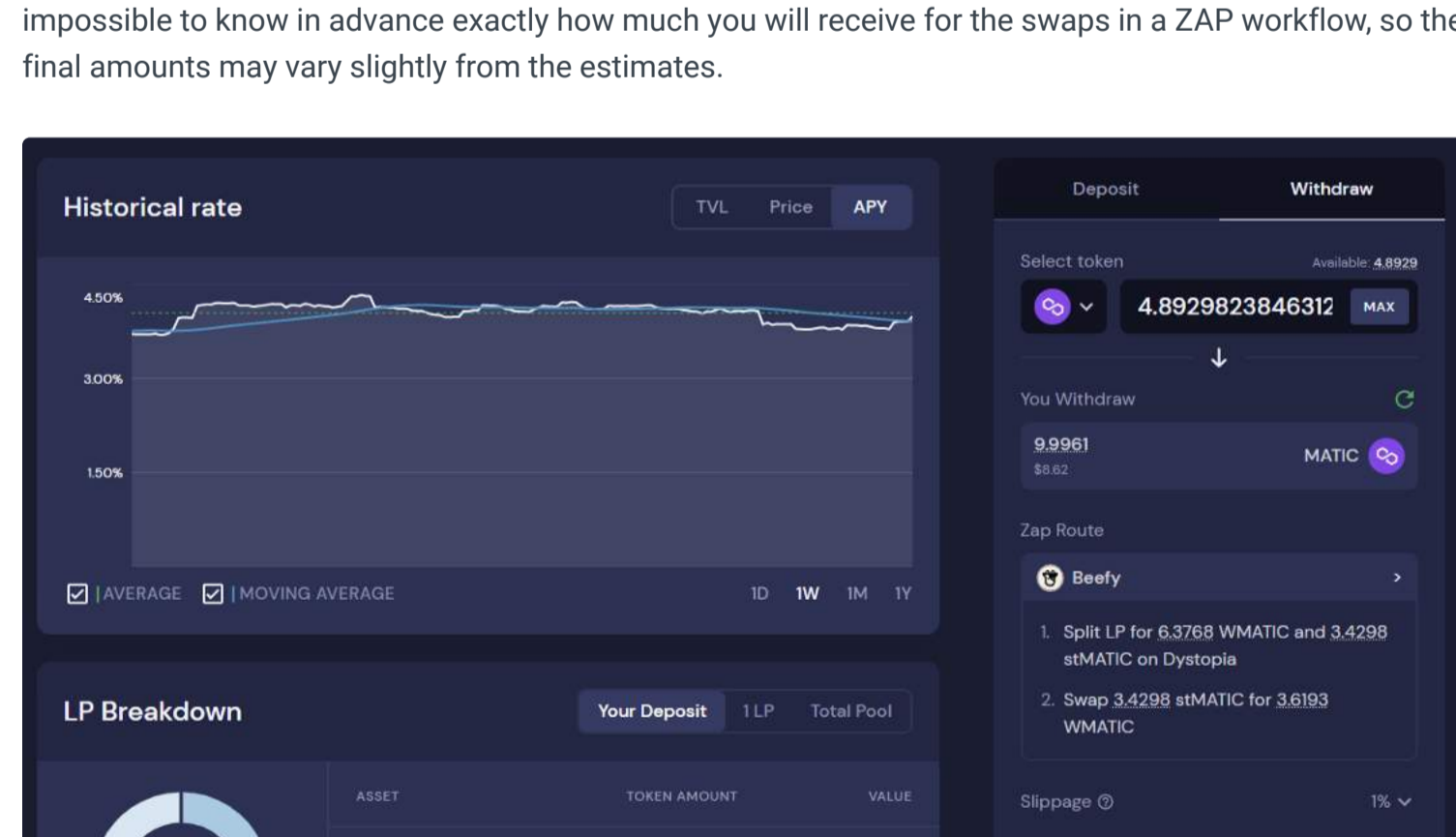
To execute the proposed transaction, click on 'Deposit' or 'Deposit All', and your chosen wallet will trigger a transaction (possibly including an approval transaction first, if your allowance for the chosen asset is insufficient). Once you've executed the transaction, you will receive confirmation that the transaction has been submitted to the blockchain and that confirmation is pending. Once the transaction is validated on the blockchain, you will then receive confirmation that the transaction was completed, meaning the vault tokens will now be in your wallet.

And that's it! Deposit complete. Beefy ZAP V1 automatically created liquidity with Dystopia and deposited that liquidity into the Beefy vault. You can check [this guide](#) to see when our vaults will harvest rewards and compound for more LP tokens.

## ZAP V1: Exiting Vaults

In the reverse, you can withdraw by selecting the 'Withdraw' tab instead of 'Deposit'. Use the 'Select token' dropdown to select one of the underlying assets of the vault (WMATIC, stMATIC, MATIC). You can then select the quantity of LP tokens that you wish to withdraw, or use the 'MAX' button to select all. Please note that the number of LP tokens is ordinarily different to the number of main tokens or the number of underlying asset tokens you will receive, so the figure displayed can cause confusion.

The UI will then display an estimate for how much of the chosen withdrawal token (e.g. MATIC) you will receive through the ZAP V1 workflow. Below, the 'Zap Route' box shows the different steps and estimates for how much of each token will be received and used in each subtransaction. As with deposits, it is impossible to know in advance exactly how much you will receive for the swaps in a ZAP workflow, so the final amounts may vary slightly from the estimates.



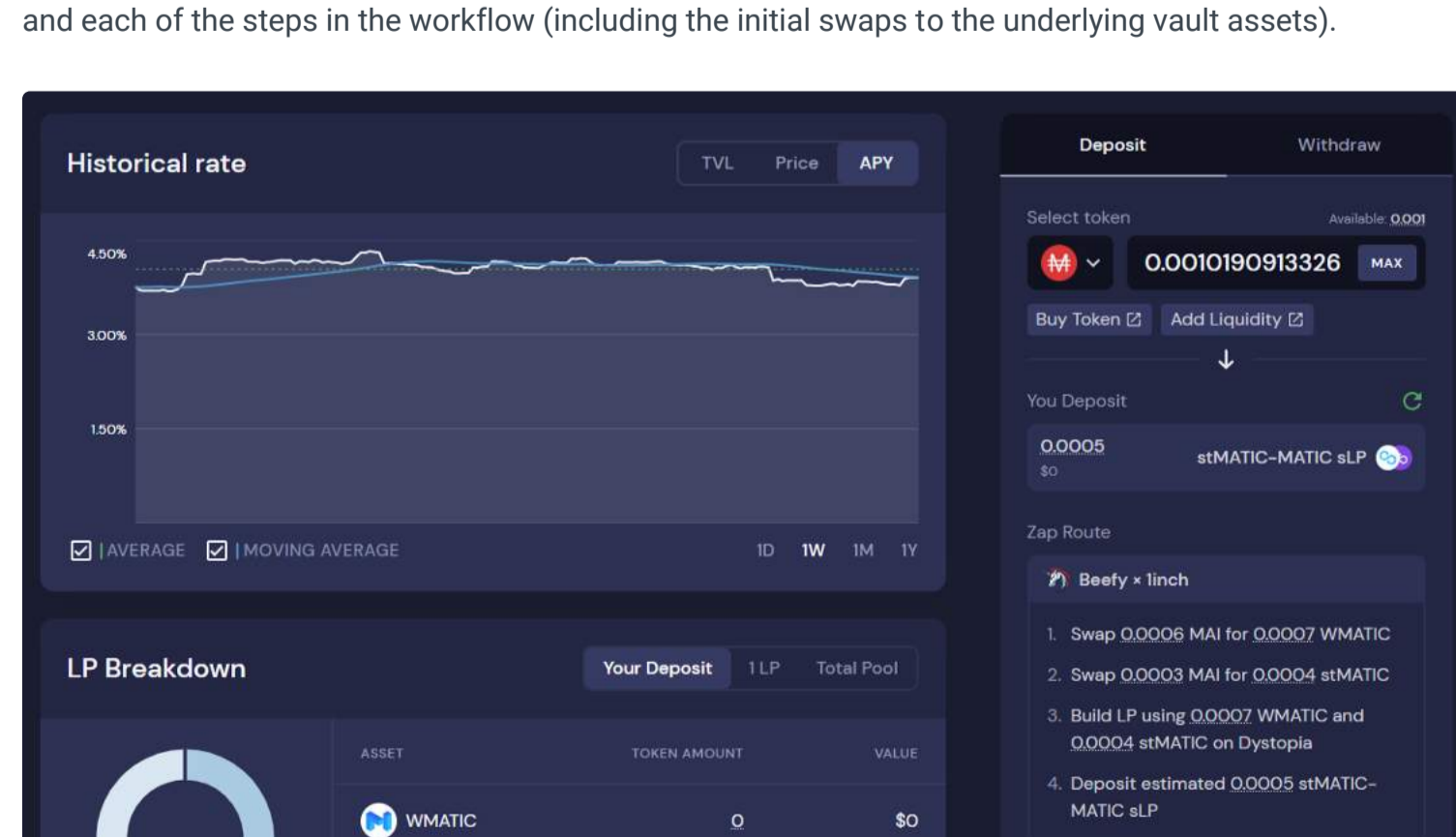
For the ZAP V1 withdrawal workflow, select one of the underlying assets in the vault and input the number of LP tokens you wish to withdraw from the vault.

To execute the proposed transaction, click on 'Withdraw' or 'Withdraw All', and your wallet will trigger another transaction (including approvals if needed). Once you've executed the transaction, you will receive confirmation that the transaction has been submitted to the blockchain and confirmation is pending. Once the transaction is validated on the blockchain, you will then receive confirmation that the transaction was completed, meaning the withdrawal tokens will now be in your wallet.

There you go! Withdrawal complete. Beefy ZAP V1 took all the steps to exit the vault, break the liquidity position and swap back to your chosen asset.

## ZAP V2: Entering Vaults

The ZAP V2 workflow is nearly identical to V1 from a user perspective. You can select any of the available tokens in the 'Select token' dropdown menu, and input the amount to deposit in the box to the right. Again the UI will show you estimates to the number of LP tokens you will receive for depositing into the vault, and each of the steps in the workflow (including the initial swaps to the underlying vault assets).

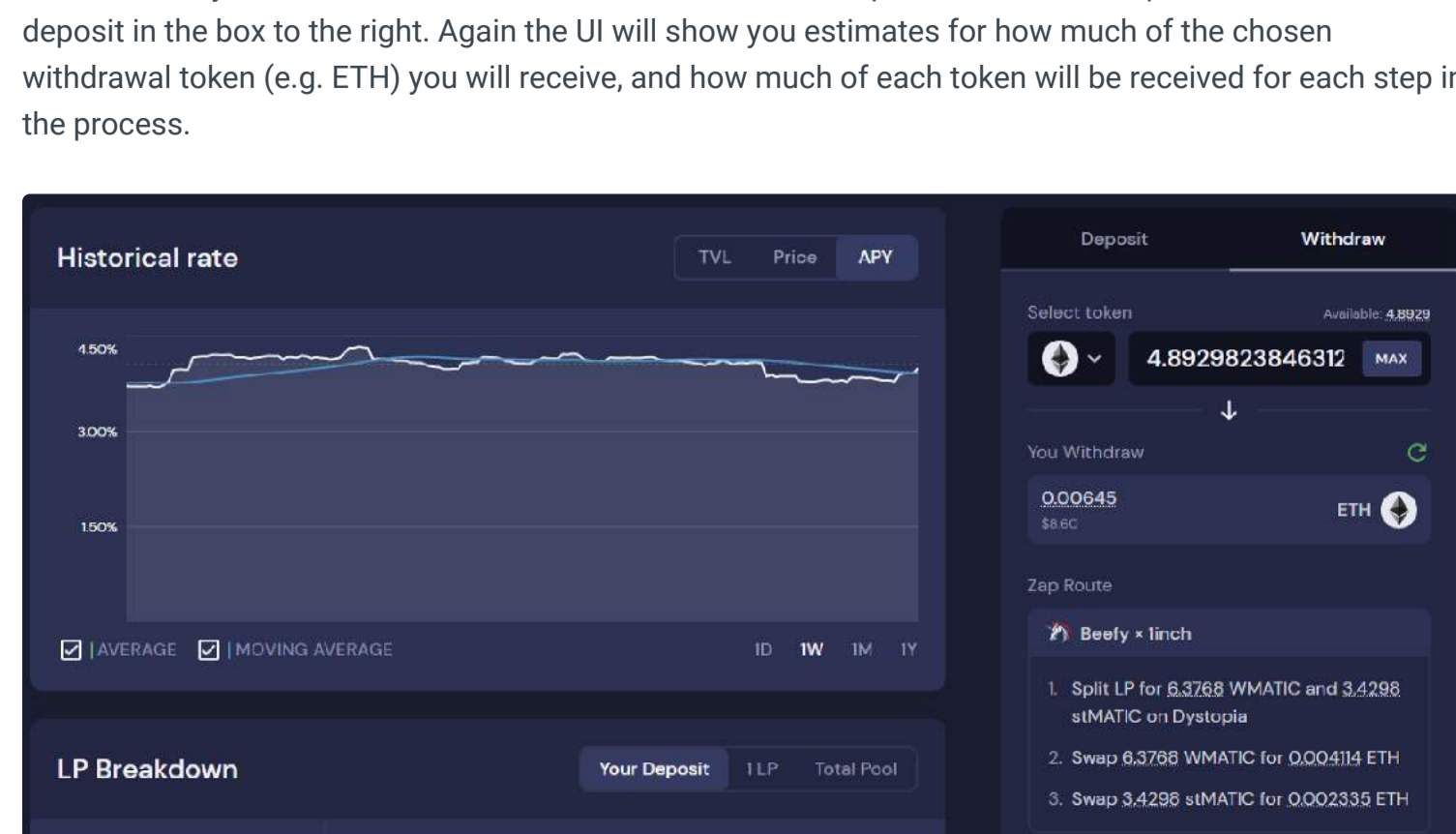


For the ZAP V2 deposit workflow, select any of the available assets in the dropdown menu and input the quantity of that token you wish to deposit into the vault.

As with the V1 process detailed above, simply hit 'Deposit' or 'Deposit All', run through the transaction on your wallet, and watch the transaction complete on the blockchain. Deposit complete! Beefy ZAP V2 took you from your chosen assets into the underlying vault assets, and then ran the same workflow as ZAP V1.

## ZAP V2: Exiting Vaults

Similarly, the ZAP V2 withdrawal workflow is also nearly identical to V1 for users. As with deposits, you can now select any of the available tokens in the 'Select token' dropdown menu, and input the amount to deposit in the box to the right. Again the UI will show you estimates for how much of the chosen withdrawal token (e.g. ETH) you will receive, and how much of each token will be received for each step in the process.



For the ZAP V2 withdrawal workflow, select any of the available assets in the dropdown menu and input the number of LP tokens you wish to withdraw from the vault.

As with the V1 process detailed above, simply hit 'Withdraw' or 'Withdraw All', run through the transaction on your wallet, and watch the transaction complete on the blockchain. Withdraw complete! Beefy ZAP V2 took you through all the steps in V1 to exit the vault and break the liquidity position, before swapping back to your chosen asset with V2.

The ZAP quote that's displayed during the deposit or withdraw process always has the ZAP fee taken into account. Additionally, the ZAP fee is only deducted from token swaps. The fees first accumulate in a batch treasury and after some time are swapped to stables and sent to Beefy's Treasury. The original Beefy ZAP V1 remains free to use.

# How to add and switch networks on Beefy

In this guide new Metamask network settings will be added for you using the Beefy network switcher.

As Beefy is a Decentralized, Multi-Chain Yield Optimizer, users will need to have properly configured wallets for each chain that they want use Beefy on. Using this guide and the Beefy network switcher, new blockchain networks can be added automatically with ease. As an example the Huobi ECO chain (HECO) network will be used in this How-To guide, but you can repeat the same process for any other network listed in the network switcher.

## Prerequisites

- A Binance Smart Chain (BSC) configured Metamask wallet is needed. Beefy started on BSC so you probably already have this set up, but if you do not have a configured wallet you can use the following guide from Binance Academy: [Connecting Metamask to Binance Smart Chain](#) (you may skip the testnet section, it is not needed).

## Walkthrough

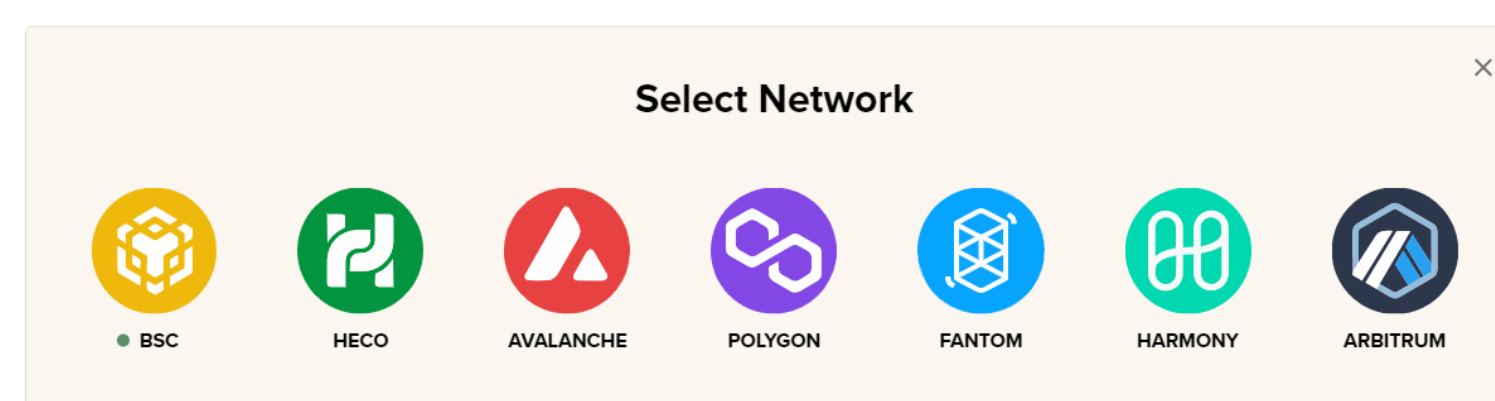
1. Go to [app.beefy.finance](https://app.beefy.finance) and connect your BSC configured Metamask wallet.



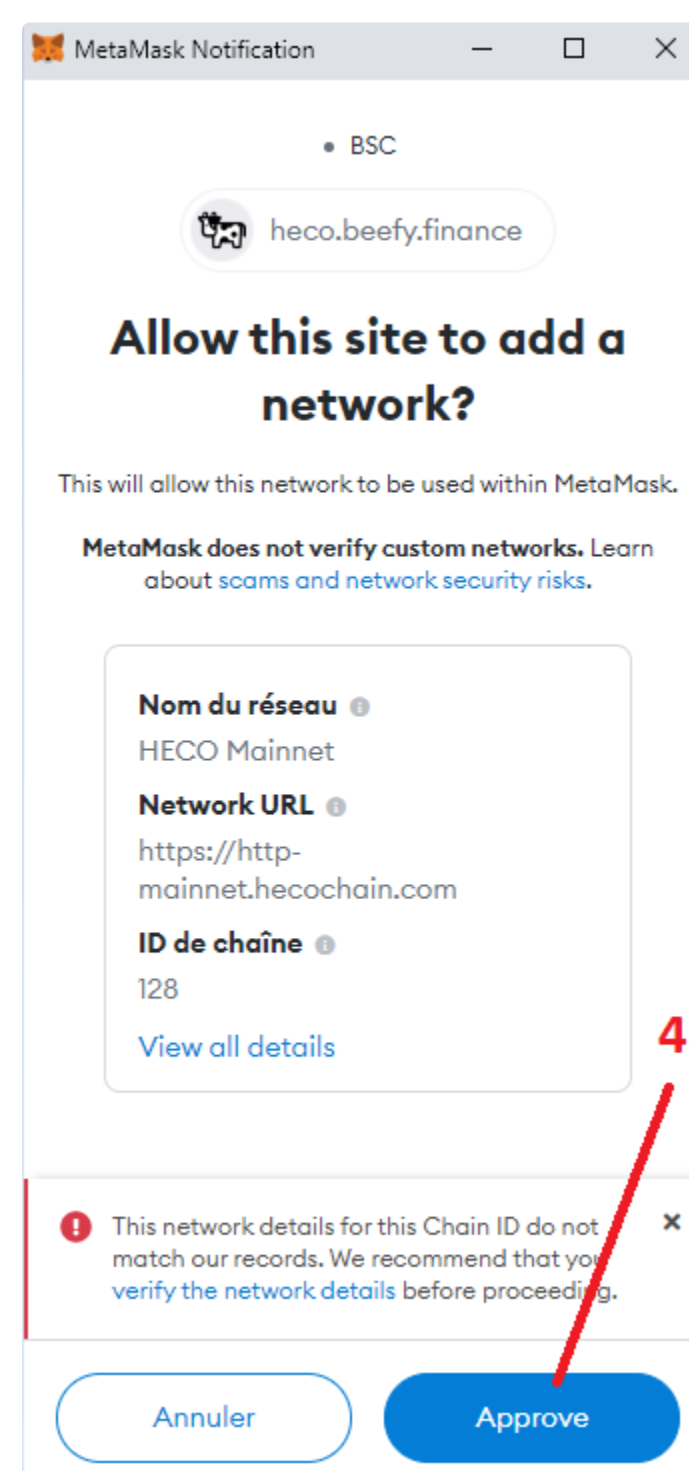
2. Click on the BSC icon under 'Select Network'.



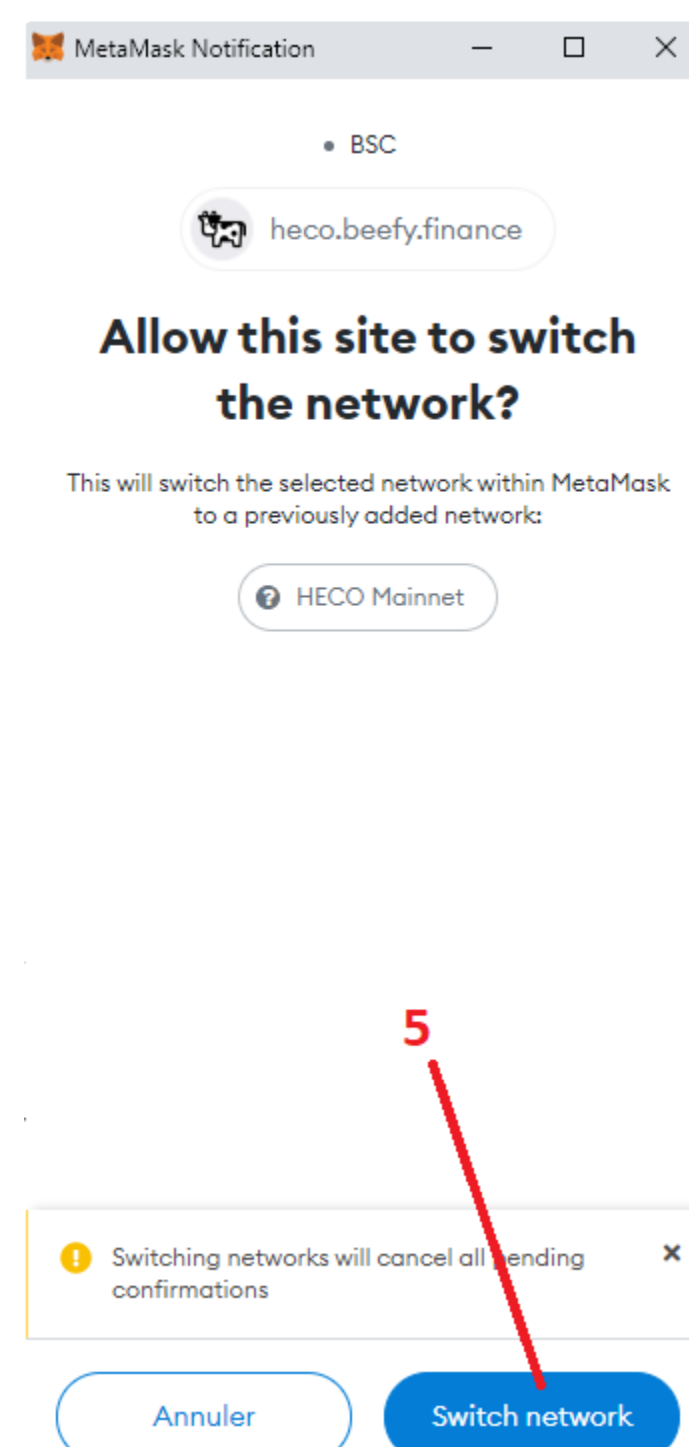
3. Select HECO (or any other network you want to add).



4. Approve new network creation to your Metamask



5. Allow Beefy to switch to the HECO network



And that's it! HECO is now configured and added to your Metamask wallet. You can repeat the process for any other network that you want to add or want to switch to.

# How to bridge BIFI cross-chain

In this guide you will find the required steps to bridge BIFI cross-chain using Metamask and AnySwap.

As a true multichain token, BIFI can be staked to earn a share of Beefy's revenue on every chain integrated by Beefy. Each chain holds different opportunities for staking your BIFI, just check out the different APYs of the chain's BIFI Maxi vault!

## Prerequisites

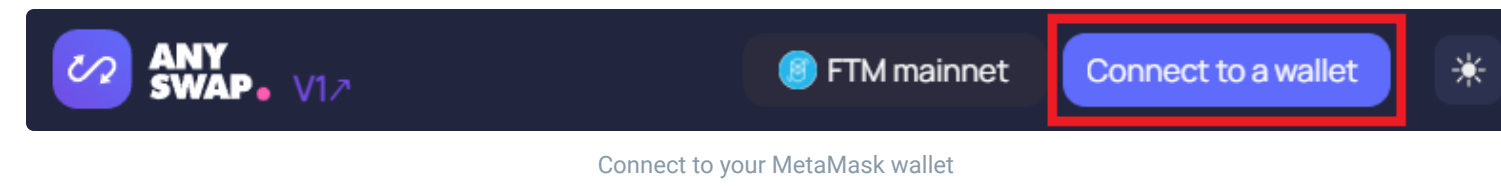
- You need to have the current and destination network configured in your Metamask.
- You will need the current and destination chain's native gas token to pay for transaction fees.

## Walkthrough

You may bridge BIFI from any of the support networks. As an example, we are going to bridge BIFI from Fantom to Cronos in this guide.

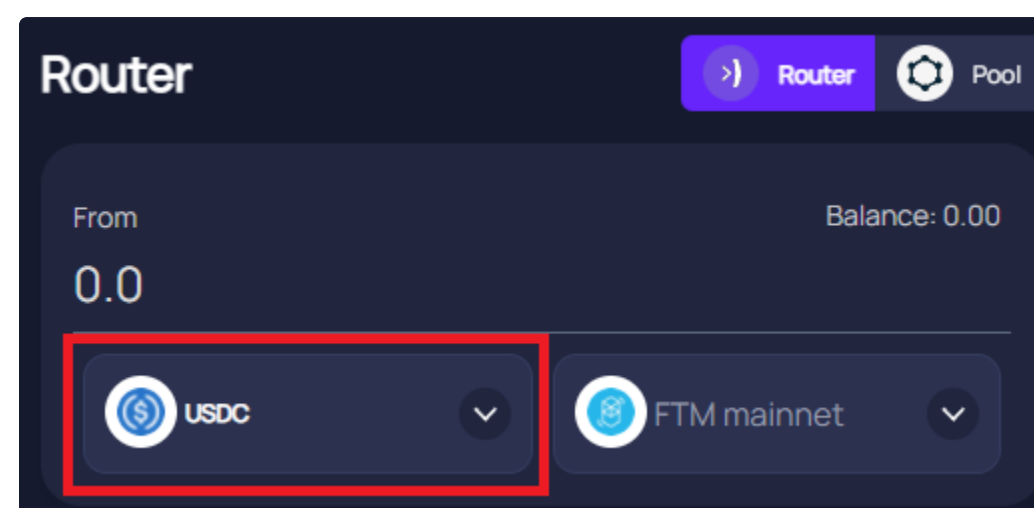
1. Visit <https://anyswap.exchange/#/router>

2. Switch network to Fantom and connect your wallet

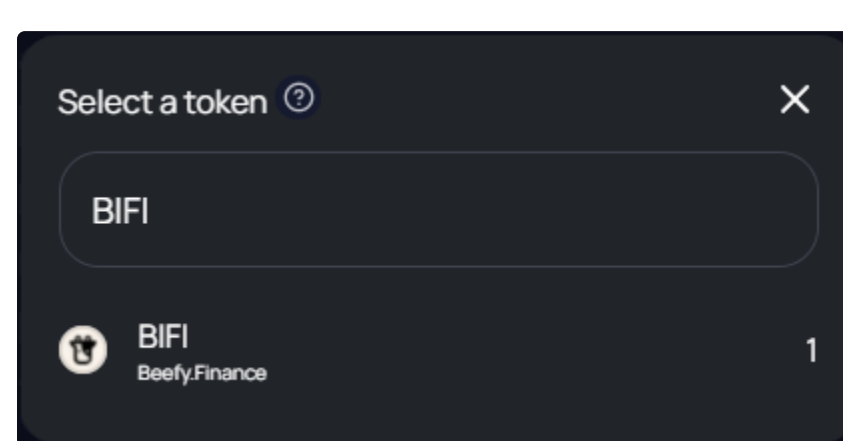


3. Select BIFI to send

Click the USDC icon in the "From" field,

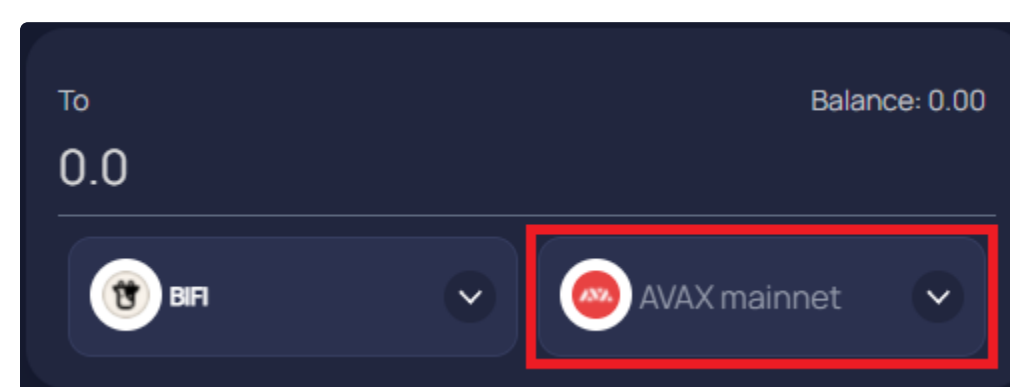


and select BIFI:

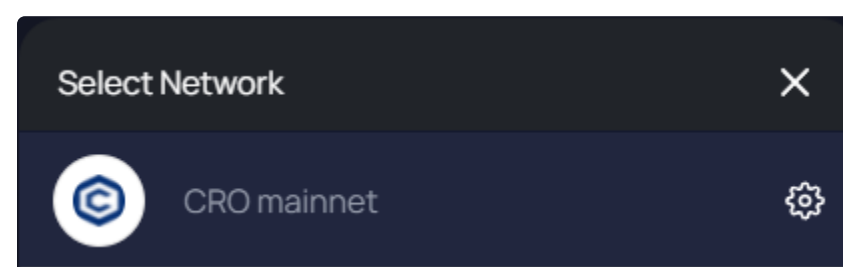


4. Select BIFI to receive

At this step, you can select on which network you wish to receive BIFI. Click on the destination chain:

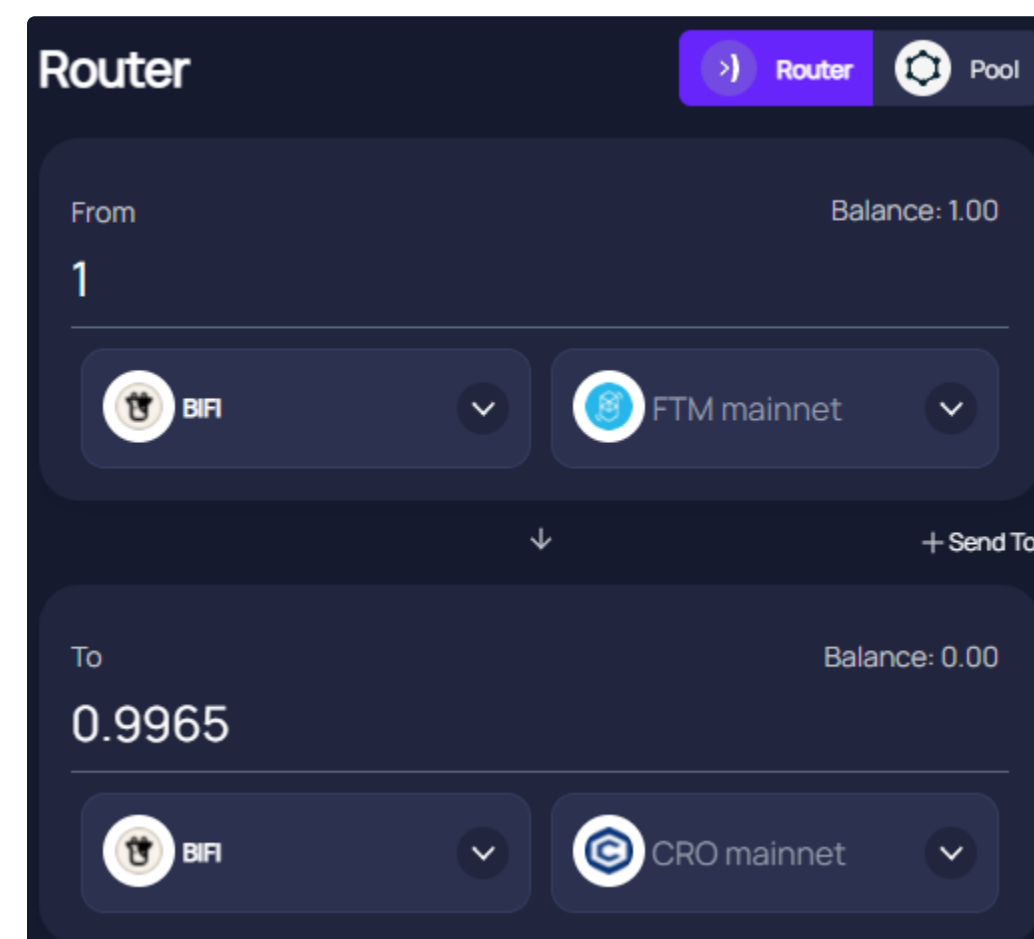


We select Cronos in this guide:

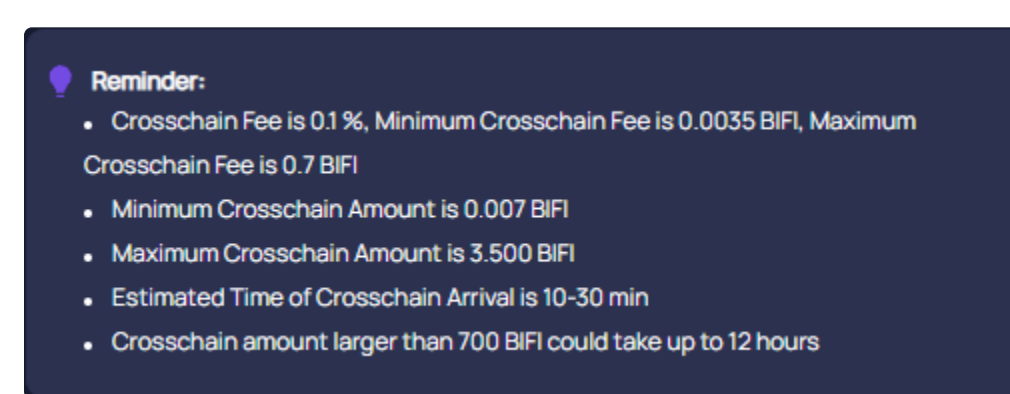


5. Specify the amount to send

Either type in the amount, or click your balance.



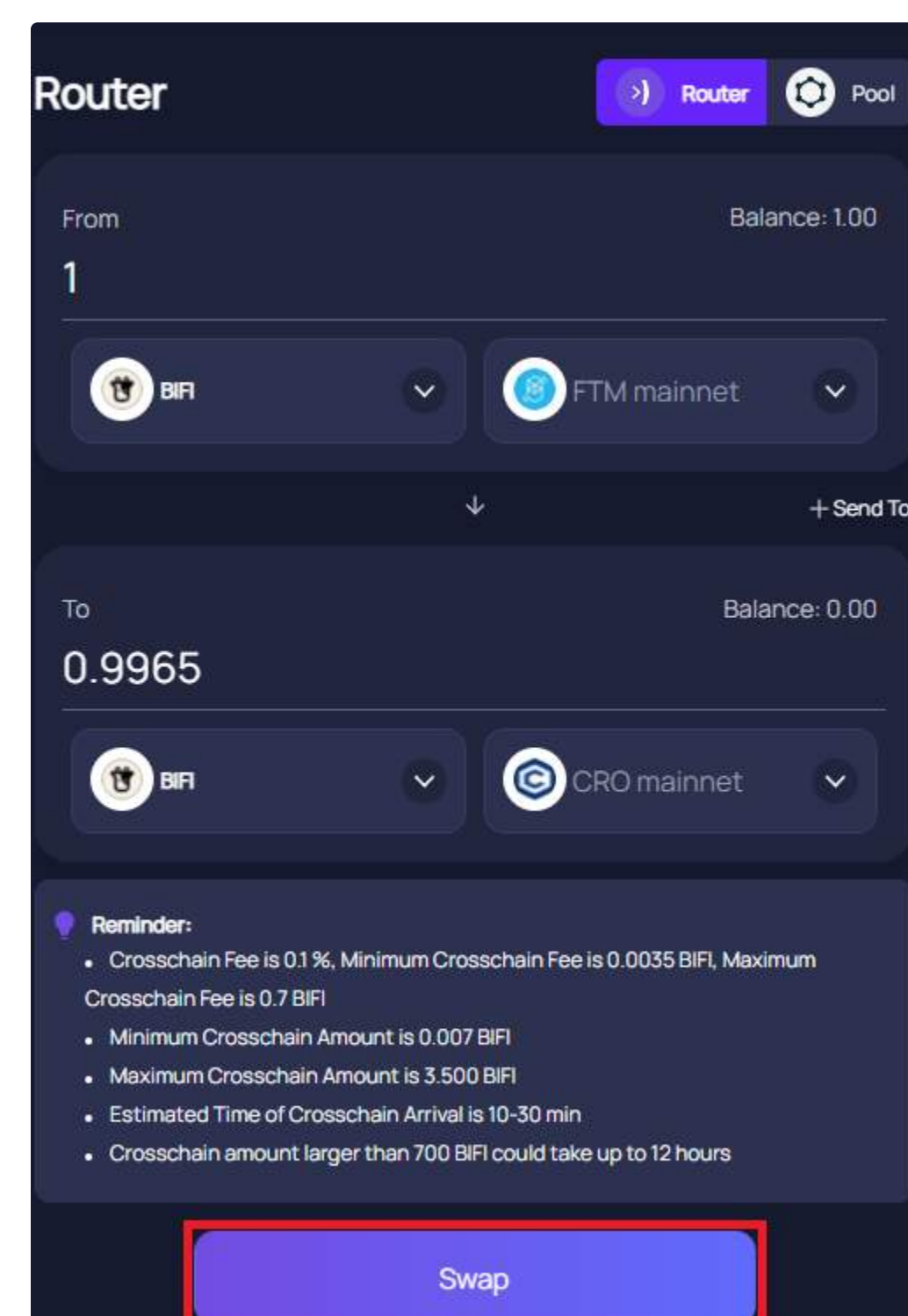
6. Read the terms of service



AnySwap's terms of service as of December 2021

6. Send it!

If you consent with the terms of service, click the "Swap" button.



That's it! Your BIFI will now be sent to the Cronos network! 🎉

# How to check the harvesting and compounding rate of a vault

In this guide you will find the required steps to check a vault's harvesting and compounding rate.

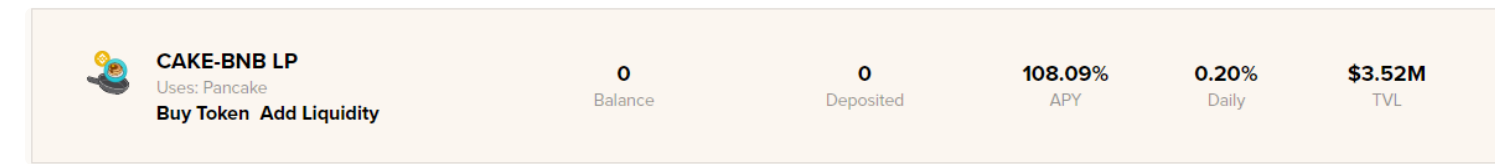
Beefy's vaults, or more specifically the investment strategy tied to the vault, will automatically increase the amount of your deposited asset by compounding arbitrary yield-farm rewards back into more of that asset. This constant cycle of harvesting rewards, and compounding, happens usually multiple times per day. In this How-To we walk you through steps to check precisely how often the compounding occurs.

## Walkthrough

NOTE: No matter which chain you choose, you can use Beefy's [dashboard.beefy.finance](https://dashboard.beefy.finance) to launch your investigation.

### Binance Smart Chain (BSC) example

Let's choose the CAKE-BNB LP vault on the Binance Smart Chain to demonstrate:



Screenshot taken 5 May 2021

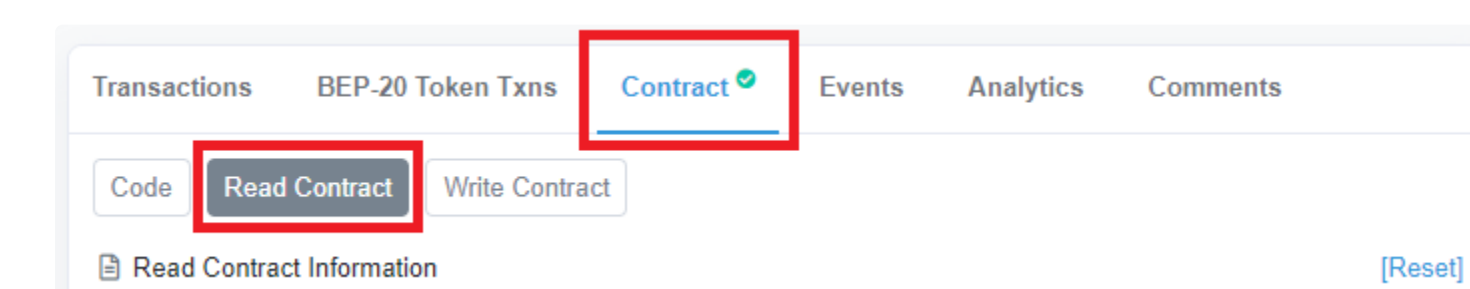
#### 1. Go to [dashboard.beefy.finance](https://dashboard.beefy.finance)

This dashboard chooses which statistics and vaults to show based on the blockchain network your wallet (e.g. MetaMask) is connected to. So if it's not now on BSC, simply switch networks to that and the dashboard page will refresh to display Beefy's statistics and vaults on BSC.

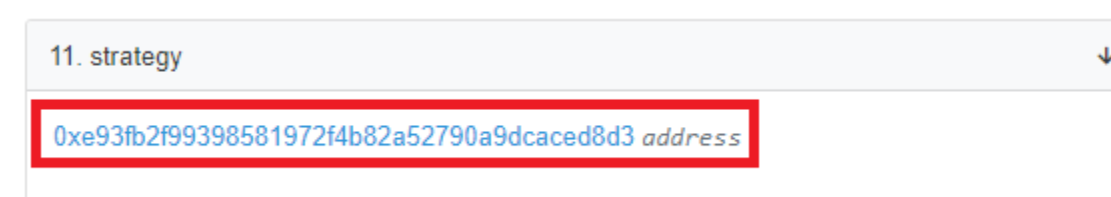
#### 2. Find the contract for the vault you wish to inspect, and click on it, opening a page in the BscScan block explorer



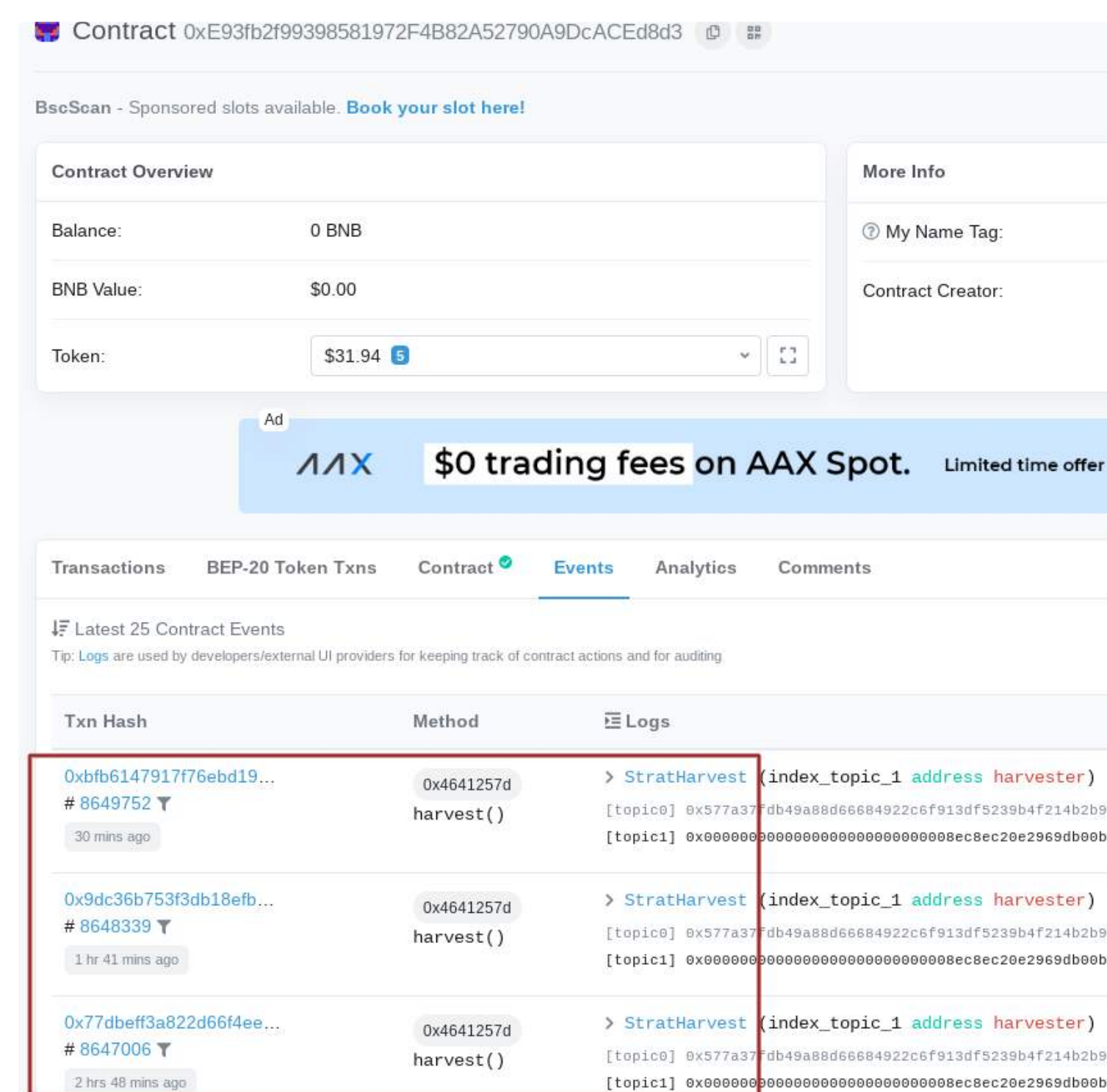
#### 3. On the BscScan page, open the "Contract" tab and subsequently the "Read Contract" tab



#### 4. Scroll down to find the strategy contract, and click on it



#### 5. Click on the Events tab to view the strategy events that have fired



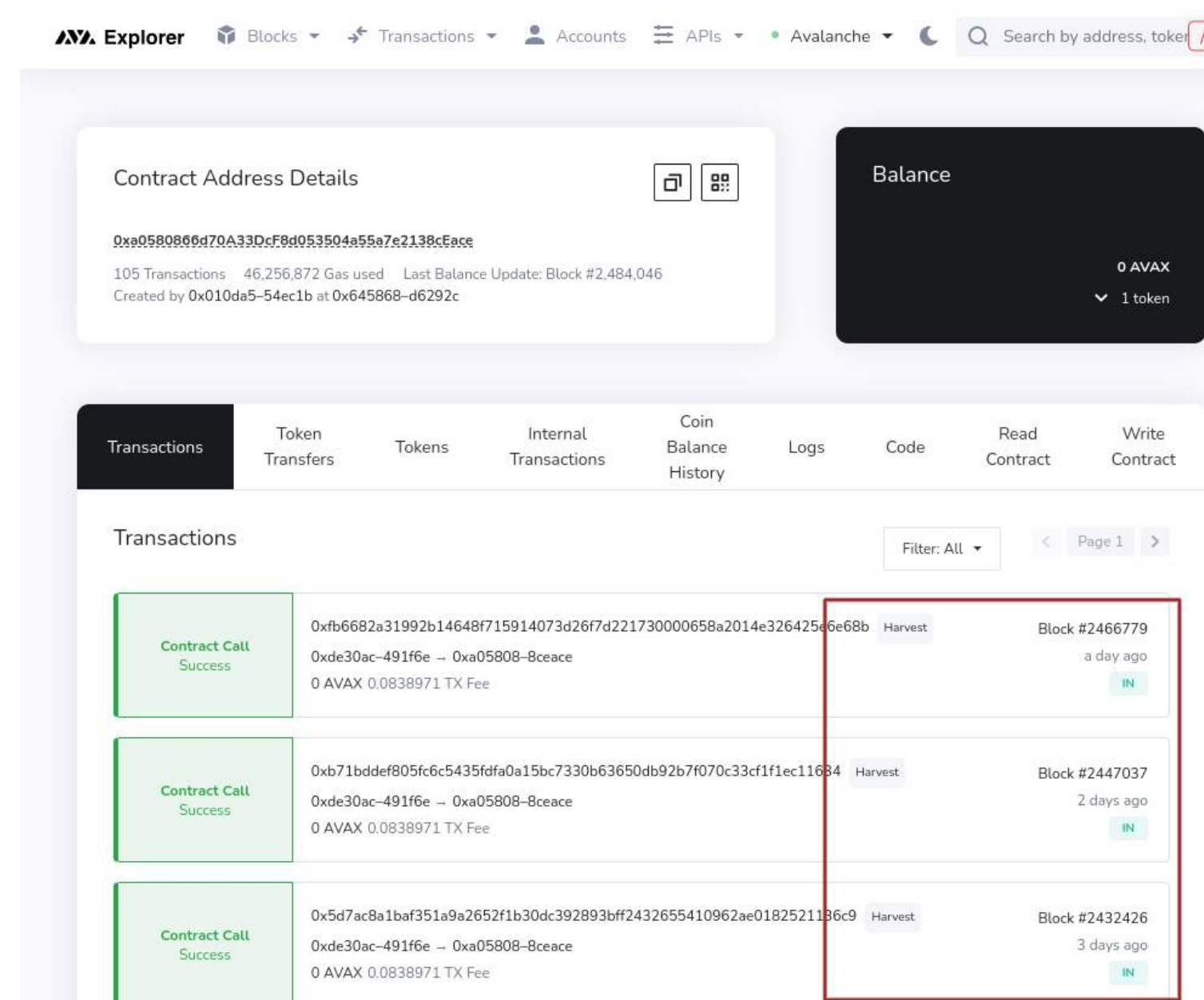
The "StratHarvest" events are where LP-farming rewards are culled and in turn compounded into more of the underlying LPs, the initial deposited asset, and then redeposited into the Beefy vault. As the timestamps reflect, this CAKE-BNB vault compounds roughly once per hour.

### Other Chains (except Harmony, Celo, Cronos)

Each of the chains supported by Beefy may be investigated via the same method shown above for BSC. The only difference will be the block explorer opened. For example on Polygon, PolygonScan will open.

### Harmony, Celo, Cronos

The basic method shown in the BSC example above still applies, except for the last, key step 5. This owes to these chains using different block-explorer softwares. In these cases, step 5 switches to the Transactions tab to view the strategy events fired, as exemplified below

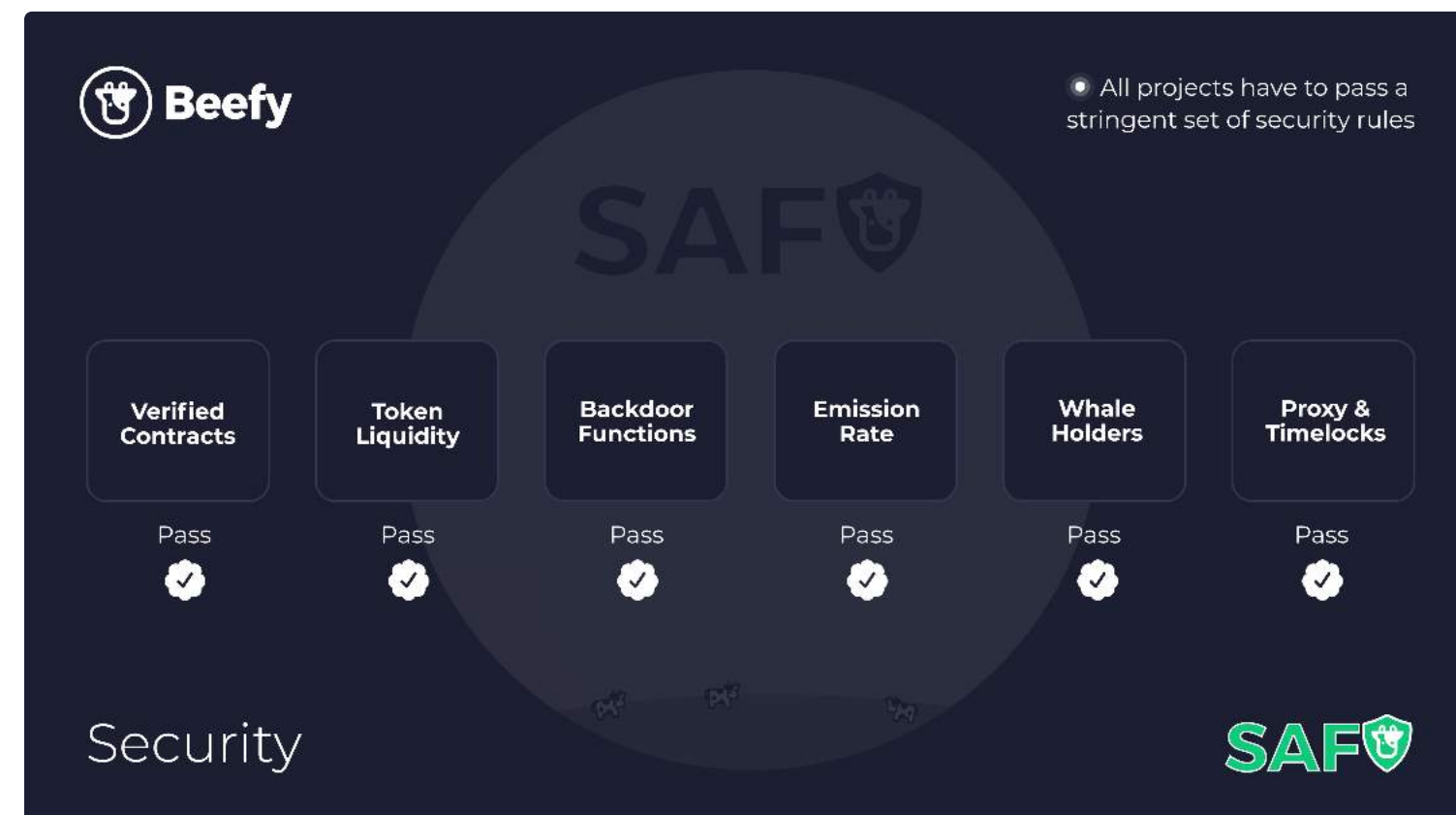


# Beefy SAFU Practices

"Don't Trust, Verify"

At Beefy, we recognise three words to live by: *SAFU First. Always.* You can craft the most incredible features into your smart contracts, but if you can't adequately safeguard user funds, your contracts don't deserve to have users. That's why safety is the first, last and foremost consideration in every product we release.

This page describes the various practices that Beefy's contributors take to ensure that all new products are safe before launch, all ongoing products are properly maintained and observed, and that our response to any security concerns are issues are rapid and safe.



Beefy has adopted a stringent SAFU management approach, that monitors a range of security factors as prerequisites for all of our products.

## New Farm Requirements

Before a new farm can be vaulted on Beefy, the projects involved with the farm have to pass a stringent set of SAFU rules:

- contracts must have been verified in the block explorer;
- non-native tokens must be from reputable bridges;
- liquidity must be sufficient for swapping farm token rewards;
- rug/migrator functions must be either completely removed or timelocked sufficiently;
- farm token emission rates must have been timelocked (if farm token pairs are being vaulted);
- farm token holders with >5% circulating supply must not be either externally owned accounts ("EOAs") or multi-sigs; and
- all proxy implementation changes (i.e. upgrades to the contracts) must be timelocked.

## New Vault Testing Procedure

Once a farm has been accepted and a vault is being prepared, our strategists will follow a manual testing procedure on every new vault before it can go live on our app. This is to ensure that the vault works as intended and user funds are always SAFU. The sequential procedure is:

1. deposit a small amount of the asset;
2. withdraw all;
3. deposit again, wait 1 minute and check that `callReward()` is not 0;
4. harvest the strategy;
5. panic the strategy;
6. withdraw 50% while panicked to make sure users can leave;
7. try to deposit, an error should pop up but don't send the deposit through;
8. unpause the strategy; and then
9. deposit the 50% that has previously been withdrawn and harvest again.

## Strategy upgrades

Occasionally, Beefy strategists will come out with a new innovative strategy, or yield farms change their reward contracts. If that's the case, Beefy vaults have the flexibility to adapt to these changes, and have the ability to swap strategies so users don't have to migrate their funds to a new vault: it's done automatically by a strategy upgrade.

The new strategy is deployed with a dummy vault and all of the manual tests outlined above are completed. After passing the checks, the new strategy is assigned to the vault. The vault gets proposed the new strategy through a multi-sig wallet and has to wait until the timelock delay has passed before the vault uses the new strategy.

## Timelock Monitor

During the life of our vaults, the projects and protocols that we build on top of will naturally need to use functionality in their smart contracts which are susceptible to abuse (and for which a timelock was required to implement a Beefy vault). To protect against the risk of abuse, we have implemented the `#timelock-monitor` channel on the [Beefy Discord](#).

By displaying the relevant contract and protocol, the triggered event, the method scheduled to be called and the end time for the timelock, the Timelock Monitor provides all the information needed to assess the risk and protect user funds. As a public channel, the monitor provides free access to this information for our users, as well as insight into how Beefy's team is handling these risks, to inform decision-making.

## Panic

Even with all of our precautions, sometimes something can go wrong with the underlying farm or assets in a Beefy vault, for which reacting quickly is of great importance. Beefy strategies have a keeper that is allowed to panic, which withdraws the staked funds from the farm back to the strategy contract and removes all allowances. This ensures that funds are always available for Beefy stakers to withdraw in case of emergency.

# Contracts & Timelocks

:

## Address Book

All addresses used are open source and verifiable. A collection of useful addresses on Beefy's chains for DeFi development are stored on GitHub: [Address Book](#).

## Contracts

From the Vault UI, one can easily find the Strategy addresses and Vault addresses. Additionally, all Beefy vault contracts can be viewed on [dashboard.beefy.finance](#). One can use this dashboard for example to check the [harvesting and compounding rate](#) of a vault.

## Oracles

Beefy's contracts do **not** use external oracles. The problem with oracles is, in short, that its data can be inaccurate or manipulated, and unreliable oracles can lead to exploits. Because Beefy's contracts do **not** rely on external data in any form, such as asset prices, our vaults are **not** susceptible to flashloan exploits.

## Example contracts

- DAI/USDC/USDT (Curve - Avalanche) vault code:  
<https://snowtrace.io/address/0x79A44dc13e5863Cf4AB36ab13e038A5F16861Abc#code>
- WBTC (Scream - Fantom) lending strategy code:  
<https://ftmscan.com/address/0x4374207377C1A36e386A757B774D53a0B6Ff2cEE#code>
- CAKE-BNB (PancakeSwap - BNB Chain) regular strategy code:  
<https://bscscan.com/address/0xDE238C509bcCBCd91B90dE40dF3e25B43A131311#code>

## Timelocks

Contracts are secured with timelocks and multi-sig dev wallets. A 6 hour timelock is used for agility to make needed changes to keep our contracts secure, and as an added layer of protection the timelock is governed by a 3/5 signer multisig.

- Arbitrum (6 hours): 0x6d28afD25a1FBC5409B1BeFFf6AEfEEe2902D89F
- Avalanche (6 hours): 0x37DC61A76113E7840d4A8F1c1B799cC9ac5Aa854
- BSC (6 hours): 0x65CF7E8C0d431f59787D07Fa1A9f8725bbC33F7E
- Celo (6 hours): 0x5B96bbAca98D777cb736dd89A519015315E00D02
- Cronos (6 hours): 0x4f4DB83d75876f34fd927d5fa78D5D7b4479E6ce
- Fantom (6 hours): 0x847298aC8C28A9D66859E750456b92C2A67b876D
- Fuse (6 hours): 0xa9E6E271b27b20F65394914f8784B3B860dBd259
- HECO (6 hours): 0x587479672077fBD7cb08EE1fd13fca6a9ef69d9e
- Metis (6 hours): 0xdf68Bf80D427A5827Ff2c06A9c70D407e17DC041
- Moonriver (6 hours): 0xc8BD4Ae3d3A69f0d75e3788d2ee557E66EBC98D8
- Harmony One (6 hours): 0x6d28afD25a1FBC5409B1BeFFf6AEfEEe2902D89F
- Polygon (6 hours): 0x6fd13191539e0e13B381e1a3770F28D96705ce91

## Developer Multisigs

Multi-signature developer wallets are used to deploy changes to contracts, such as upgrading vault strategies. This ensures a secure workflow where every change is approved by Beefy's developers.

- Arbitrum: 0xf7EC8986c660Fa8269f6440A631B22337f398Ccd
- Avalanche: 0x3A0b8B7a3ea8D1670e000b1Da5bD41373bF8da42
- BSC: 0x44b74ED902e6423B51Bd9e44B6e5646749376943
- Fantom: 0x238dc3781DD668abd5135e233e395885657D304A
- Fuse: 0xe26a8aC2936F338F4DAebA4BD22a7ec86465fE1
- Harmony: 0xE3c985f5e317eFd4aca1f00aa5F1DFEC40b2Af74
- Polygon: 0x09dc95959978800E57464E962724a34Bb4Ac1253
- Metis: 0xF9810A3dA8a554B84Ed79D67461eCA6Eb3fA9BD
- Moonriver: 0x1fdd00b45eba7f6d35b92803eadd68f7cc4a193

## Treasury Multisigs

Beefy's treasury spending is secured by requiring multiple signatures from trusted (community) members. As voted on by the DAO, the following members represent the Treasury Council: Power, AllTrades, Pablo, mjoaris, TBC, DefiDebauchery and YR2150. See the [Treasury](#) page for further information about the Beefy Treasury.

- Arbitrum: 0x3f5eddad52C665A4AA011cd11A21E1d5107d7862
- Aurora: 0x088C70Ddff3a3774825dd5e5EaDB356404248d83
- Avalanche: 0x26dE4EBffBE8d3d632A292c972E3594eFc2eCeEd
- BSC: 0x7C780b8A63eE9B7d0F985E8a922Be38a1F7B2141
- Canto: 0xF09d213EE8a8B159C884b276b86E08E26B3bf75
- Celo: 0xca807d809f9639cefb3d31a7951cec3ab405a2fd
- Cronos: 0xa9721Ae5042482D7a884A2138f580459B680920f
- Ethereum: 0xc9C61194682a3A5f56BF9Cd5B59EE63028aB6041
- Emerald: 0x8fd0869271d26e6653f5d5650685630f75b6aefd
- Fantom: 0xdFf234670038dEfb2115Cf103F86dA5fB7CfD2D2
- Fuse: 0x1C124c2CaB83b3C3B5D0f0899CeeA5e06964599F
- Harmony: 0x523154a03180FD1CB26F39087441c9F91BcD0389
- HECO : 0xdbb72c8b7ebdd52a4813b9d262386dfdab69c9ba
- Kava: 0x07F29FE11FbC17876D9376E3CD6F2112e81feA6F
- Metis: 0x0f9602B7E7146a9BaE16dB948281BebDb7C2D095
- Moonbeam: 0x3E7F60B442CEAE0FE5e48e07EB85Cfb1Ed60e81A
- Moonriver: 0x617f12E04097F16e73934e84f35175a1B8196551
- Optimism: 0x4ABa01FB8E1f6BFE80c56Deb367f19F35Df0f4eE
- Polygon: 0xe37dD9A535c1D3c9fC33e3295B7e08bD1C42218D

# Bug Bounty Program

:

Hunt bugs and get paid!

Our developers typically conduct rigorous internal tests and evaluate blockchain contracts before deployment. Beefy has also been **audited** by security organisation **CertiK**. Furthermore, the more widely available the source code is for public testing, scrutiny, and experimentation, the more rapidly all forms of bugs will be discovered. "Given enough eyeballs, all bugs are shallow." - this is a key Beefy principle.

However, in order to guarantee a top level security for your deposits, one can never be too careful. That's why Beefy has launched a bug bounty program with ImmuneFi. On Immunefi, white hat hackers secure DeFi contracts, save funds from theft, and get paid doing it.

The bug bounty program covers Beefy's smart contracts and apps and is focused on the prevention of the following:

- Significant Vault hack/exploit;
- Theft of Governance Funds;
- Website down/DDOS attack.

More details [here](#).



# Beefy Safety Score

This document outlines the design for the Beefy Safety Score. The purpose of the safety score is to educate users when making a decision to enter a particular Beefy vault. The Safety Score is not necessarily perfect, but it is another tool that helps the user.

The safety score that a vault can get goes from 0 to 10. The best possible score is 10 and the worst is 0. It is technically possible for vaults to score less than 0, in which case 0 will be displayed.

Risks are distributed in three main categories:

- Beefy Risks: Risks that we add by serving as a platform.
- Asset Risks: Risks of the asset being handled by the vault.
- Platform Risks: Risks of the underlying farm or platform used.

Each category is responsible for a percentage of the total score. Each category is itself divided in multiple subcategories.

All vaults start with a perfect score of 10 and are subtracted points whenever they have qualities that increase risk.

## Category: Beefy Risks

These are risks related to the Beefy platform itself. These could be risks added by the complexity of the vault strategy, if it's an experimental deployment, if it's been audited by others, etc. Twenty percent of the safety score is determined by the Beefy Risks.

### Subcategory: Complexity

Tracks the complexity of the strategy behind a vault.

#### COMPLEXITY\_LOW

- Title: Low complexity strategy
- Explanation: Low complexity strategies have few, if any, moving parts and their code is easy to read and debug. There is a direct correlation between code complexity and implicit risk. A simple strategy effectively mitigates implementation risks.
- Qualification Criteria: A low complexity strategy should interact with just one audited and well-known smart contract e.g. MasterChef. The strategy serves as a façade for this smart contract, forwarding deposit, harvest and withdrawal calls using a single line of code.

#### COMPLEXITY\_MID

- Title: Beefy strategy is of medium complexity
- Explanation: Medium complexity strategies interact with two or more audited and well-known smart contracts. Its code is still easy to read, test and debug. It mitigates most implementation risks by keeping things simple, however the interactions between 2 or more systems add a layer of complexity.
- Qualification Criteria: A medium complexity strategy interacts with 2 or more well-known smart contracts. This strategy automates the execution of a series of steps with no forking paths. Every time deposit(), harvest() and withdraw() is called, the same execution path is followed.

#### COMPLEXITY\_HIGH

- Title: Beefy strategy is complex
- Explanation: High complexity strategies interact with one or more well-known smart contracts. These advanced strategies present branching paths of execution. In some cases multiple smart contracts are required to implement the full strategy.
- Qualification Criteria: A high level complexity strategy can be identified by one or more of the following factors: high cyclomatic complexity, interactions between two or more third-party platforms, implementation split between multiple smart contracts.

### Subcategory: Time in Market

Tracks how long has this strategy been running without any major issues.

#### BATTLE\_TESTED

- Title: Beefy strategy is battle tested
- Explanation: The more time a particular strategy is running, the more likely that any potential bugs it had have been found, and fixed. This strategy has been exposed to attacks and usage for some time already, with little to no changes. This makes it sturdier.
- Qualification Criteria:
  - Was deployed using a BeefyStratFactory
  - 10+ strategies sharing the same code deployed
  - 3 months working as expected without upgrades

#### NEW\_STRAT

- Title: Strategy has been running for less than a month
- Explanation: The more time a particular strategy is running, the more likely that any potential bugs it has have been found, and fixed. This strategy is a modification or iteration of a previous strategy. It hasn't been battle tested as much as others.
- Qualification Criteria: -

#### EXPERIMENTAL\_STRAT

- Title: The strategy has some features which are new
- Explanation: The more time a particular strategy is running, the more likely that any potential bugs it had have been found, and fixed. This strategy is brand new and has at least one experimental feature. Use it carefully at your own discretion.
- Qualification Criteria: -

## Category: Asset Risks

Risks relating to the asset or assets handled by the vault. Entering into a vault with BTC has a different set of risks than entering into a vault with a newer and smaller coin. Twenty percent of the score is determined by this category.

### Subcategory: Impermanent Loss

Tracks the risk of impermanent loss within the vault

#### IL\_NONE

- Title: Very low or zero projected IL
- Explanation: The asset in this vault has very little or even no expected impermanent loss. This might be because you are staking a single asset, or because the assets in the LP are tightly correlated like USDC-USDT or WBTC-renBTC.
- Qualification Criteria: Single asset vaults and vaults that manage stablecoins with a peg that isn't experimental. USDT, USDC, DAI, sUSD, etc.

#### IL\_LOW

- Title: Low projected IL
- Explanation: When you are providing liquidity into a token pair, for example ETH-BNB, there is a risk that those assets decouple in price. BNB could drop considerably in relation to ETH. You would lose some funds as a result, compared to just holding ETH and BNB on their own. The assets in this vault have some risks of impermanent loss.
- Qualification Criteria: Vaults that handle what are normally referred as "Pool 1" LPs would fit here: ETH-USDC, MATIC-AAVE, etc. Governance tokens for smaller projects are normally known as "Pool 2" and thereby excluded.

#### IL\_HIGH

- Title: High projected IL
- Explanation: When you are providing liquidity into a token pair, for example ETH-BNB, there is a risk that those assets decouple in price. BNB could drop considerably in relation to ETH. You would lose some funds as a result, compared to just holding ETH and BNB on their own. The assets in this vault have a high or very high risk of impermanent loss.
- Qualification Criteria: Vaults that handle "Pool 2" LPs go here. These LP normally include the governance token of the farm itself.

#### ALGO\_STABLE

- Title: Algorithmic stable, experimental peg
- Explanation: When you are providing liquidity into a token pair, for example ETH-BNB, there is a risk that those assets decouple in price. BNB could drop considerably in relation to ETH. You would lose some funds as a result, compared to just holding ETH and BNB on their own. At least one of the stablecoins held by this vault is an algorithmic stable. This means that the stable peg is experimental and highly risky. Use it carefully at your own discretion.
- Qualification Criteria: "Stablecoins" with experimental pegs, or tokenomics that have failed repeatedly to hold its peg in the past, go here.

### Subcategory: Liquidity

Tracks how difficult it is to buy/sell the vault's token.

#### LIQ\_HIGH

- Title: High trade liquidity
- Explanation: How liquid an asset is affects how risky it is to hold it. Liquid assets are traded in many places and with good volume. The asset held by this vault has high liquidity. This means that you can exchange your earnings easily in plenty of places.
- Qualification Criteria: -

#### LIQ\_LOW

- Title: Low trade liquidity
- Explanation: How liquid an asset is affects how risky it is to hold it. Liquid assets are traded in many places and with good volume. The asset held by this vault has low liquidity. This means that it isn't as easy to swap and you might incur high slippage when doing so.
- Qualification Criteria: -

### Subcategory: Market Cap

Total value of all the coins in circulation. Indirectly tracks how volatile the vault's underlying asset is.

#### MCAP\_LARGE

- Title: High market cap, low volatility asset
- Explanation: The market capitalization of the crypto asset directly affects how risky it is to hold it. Usually a small market cap implies high volatility and low liquidity. The asset held by this vault has a large market cap. This means it's potentially a highly safe asset to hold. The asset has a high potential to stick around and grow over time.
- Qualification Criteria: Top 50 MC by Gecko/CMC

#### MCAP\_MEDIUM

- Title: Medium market cap, medium volatility asset
- Explanation: The market capitalization of the crypto asset directly affects how risky it is to hold it. Usually a small market cap implies high volatility and low liquidity. The asset held by this vault has a medium market cap. This means it's potentially a safe asset to hold. The asset has potential to stick around and grow over time.
- Qualification Criteria: Between 50 and 300 MC by Gecko/CMC

#### MCAP\_SMALL

- Title: Small market cap, high volatility asset
- Explanation: The market capitalization of the crypto asset directly affects how risky it is to hold it. Usually a small market cap implies high volatility and low liquidity. The asset held by this vault has a small market cap. This means it's potentially a risky asset to hold. The asset has low potential to stick around and grow over time.
- Qualification Criteria: Between 300 and 500 MC by Gecko/CMC

#### MCAP\_MICRO

- Title: Micro market cap, Extreme volatility asset
- Explanation: The market capitalization of the crypto asset directly affects how risky it is to hold it. Usually a small market cap implies high volatility and low liquidity. The asset held by this vault has a micro market cap. This means it's potentially a highly risky asset to hold. The asset has low potential to stick around.
- Qualification Criteria: +500 MC by Gecko/CMC

### Subcategory: Supply

Tracks risks related to the asset supply. Can it be altered by anyone? How centralised is it?

#### SUPPLY\_CENTRALIZED

- Title: Few very powerful whales
- Explanation: When the supply is concentrated in a few hands, they can greatly affect the price by selling. Whales can manipulate the price of the coin. The more people that have a vested interest over a coin, the better and more organic the price action is.
- Qualification Criteria: Less than 50 accounts hold more than 50% of the supply.

## Category: Platform Risks

Risks relating to the third party platforms used by the vault. How much track record they have, how solid the code is, are there any dangerous actions that an admin can take, etc. Sixty percent of the score is determined by this category.

### Subcategory: Reputation

Tries to give clues about the team and community's track record. How likely are they to rug for example.

#### PLATFORM\_ESTABLISHED

- Title: The platform has a known track record
- Explanation: When taking part in a farm, it can be helpful to know the amount of time that the platform has been around and the degree of its reputation. The longer the track record, the more investment the team and community have behind a project. This vault farms a project that has been around for many months.
- Qualification Criteria: The underlying farm has been around for at least 3 months.

#### PLATFORM\_NEW

- Title: Platform is new with little track record
- Explanation: When taking part in a farm, it can be helpful to know the amount of time that the platform has been around and the degree of its reputation. The longer the track record, the more investment the team and community have behind a project. This vault farms a new project, with less than a few months out in the open.
- Qualification Criteria: The underlying farm has been around for less than 3 months.

### Subcategory: Security

Tracks various smart contract good practices.

#### NO\_AUDIT

- Title: The platform has never been audited by third-party trusted auditors
- Explanation: Audits are reviews of code by a group of third party developers.
- Qualification Criteria: -

#### AUDIT

- Title: The platform has an audit from at least one trusted auditor
- Explanation: Audits are reviews of code by a group of third party developers.
- Qualification Criteria: One or more audits from an auditor that has some positive track record in the space.

#### CONTRACTS\_VERIFIED

- Title: All relevant contracts are publicly verified
- Explanation: Code running in a particular contract is not public by default. Block explorers let developers verify the code behind a particular contract. This is a good practice because it lets other developers audit that the code does what it's supposed to. All the third party contracts that this vault uses are verified. This makes it less risky.
- Qualification Criteria: -

#### CONTRACTS\_UNVERIFIED

- Title: Some contracts are not verified
- Explanation: Code running in a particular contract is not public by default. Block explorers let developers verify the code behind a particular contract. This is a good practice because it lets other developers audit that the code does what it's supposed to. Some of the third party contracts that this vault uses are not verified. This means that there are certain things that the Beefy devs have not been able to inspect.
- Qualification Criteria: -

#### ADMIN\_WITH\_TIMELOCK

- Title: Dangerous functions are behind a timelock
- Explanation: Sometimes the contract owner or admin can execute certain functions that could put user funds in jeopardy. The best thing is to avoid these altogether. If they must be present, it's important to keep them behind a timelock to give proper warning before using them. This contract has certain dangerous admin functions, but they are at least behind a meaningful Timelock.
- Qualification Criteria: There is at least one function present that could partially or completely rug user funds. The function must be behind a +6h timelock.

#### ADMIN\_WITHOUT\_TIMELOCK

- Title: Dangerous functions are without a timelock
- Explanation: Sometimes the contract owner or admin can execute certain functions that could put user funds in jeopardy. The best thing is to avoid these altogether. If they must be present, it's important to keep them behind a timelock to give proper warning before using them. This contract has certain dangerous admin functions, and there is no time lock present. They can be executed at a moment's notice.
- Qualification Criteria: There is at least one function present that could partially or completely rug user funds. The function has no time lock protection.

# Token Allowance



Keep your wallet safe with Beefy Allowance.



Beefy Allowance allows you to revoke wallet access from previously Authorized Spenders.


- Site: <https://allowance.beefy.finance/>

Forked from a [token allowance checker](#) on Ethereum, this tool is used to revoke spend permissions from DApps that your wallet has previously interacted with. Leaving these spend permissions enabled could allow dishonest teams or individuals to transfer all of your approved tokens at anytime without asking further permission from you. This tool is similar to the DeBank Approval tool and [unrekt.net](#).

# Beefy Backup

:



 Both [app.beefy.com](https://app.beefy.com) and [app.beefy.finance](https://app.beefy.finance) are official Beefy websites. If one does not work for you, try the other.

Sporadically, users are experiencing white screens, or blank screens, and can not access the Beefy DApp.

This might be Fleek or Cloudflare having maintenance, but it's hard to pinpoint the root cause. When it happens, it does not affect everyone but only some regions world wide. We understand that this can be frustrating!

First and foremost, *don't worry*, your funds are always SAFU on the blockchain, they are not on Beefy's website. The site is only a front end to let you access your funds easily.

If you are experiencing the aforementioned issues, try the following official mirror: <https://beefy-v2.pages.dev>, or <https://beefy.on.fleek.co/> for the legacy UI. Using a VPN will generally help as well.

These mirror sites are a developer's build, which we can provide as a backup for users that can't access Beefy's site. Remember, seed phrases and private keys will **never** be asked.

# Insurance

Last Update: October 2022

**Disclaimer:** Nothing on this page constitutes any formal insurance, legal or financial advice, or any recommendation to purchase or not purchase the products described herein. This page is intended for general information and educational purposes only. No warranty, whether express or implied, is given in relation to the information on this page. Beefy shall not be liable to readers for any technical, editorial, typographical or other errors or omissions associated with this page, or any harm, damage, losses or claims that may result from your use of this information. Always check the date of last update, to ensure the information on this page is not materially outdated.

Though Beefy does not offer insurance coverage directly to our users, there are a number of available products which offer cover for our protocol, smart contracts and users' portfolios. This page summarises the key types of DeFi insurance products that are available at present, some of the risks associated with them, and the different Beefy products that are available right now (including from our official partners and unaffiliated products).

## Why Insurance?

On a simple level, insurance can feel like a counterintuitive concept in DeFi, as the pace of change and technology development necessarily makes DeFi an inherently risky pursuit. But digging a bit deeper, we can observe that insurance is actually one of the earliest forms of decentralised financial products, in that it seeks to socialise risk and cost among thousands of willing participants, to ensure neither is concentrated at an individual level.

At Beefy, we recognise that insurance is a requisite component of a stable, modern financial system, as it safeguards against massive damage and disruption arising from unlikely or unknowable, but retrospectively inevitable risks and events. Insurance allows users to enter the DeFi ecosystem and interact with this technology in a safer and more measured way, with protection against downside risks. It also helps to facilitate institutional capital, which may have more stringent risk management requirements that would otherwise isolate them from the space. As such, we think insurance helps to develop and extend DeFi, paving the way for mass adoption.

However, insurance is a complicated area of finance. Each and every underwritten policy is ultimately unique and deals with different types and levels of risk. As a result, it is vitally important that our users do their own research before buying insurance, and ensure their chosen products are suitable for them and priced fairly. It is always prudent to make sure to: (a) read the provider's documentation and policy documents; (b) research the provider's financial position/backing to ensure it has sufficient liquidity to cover a pay out; and (c) check to ensure the policy's coverage is appropriate for the kinds of risks you wish to ensure against.

## Types of Insurance

As DeFi expands, the list of available insurance products is growing in tandem, including from both corporate and decentralised providers. Some of the key types include:

- Smart Contract Coverage** - covers a specific smart contract-based product for a range of common issues like contract bugs, multi-sig failures and economic attacks. **For example:** [InsurAce's Smart Contract Vulnerability Cover](#).
- Protocol Coverage** - covers a specific DeFi protocol for a range of common risks and issues, like contract bugs, economic attacks and governance attacks. **For example:** [Nexus's Protocol Cover](#).
- Token Coverage** - covers a specific token product (e.g. an LP or autocompounding vault) for cross-stack risks in the underlying assets and protocols that the insured product is built on. **For example:** [Nexus's Yield Token Cover](#) (e.g. Convex 3CRV), which covers failures in the issuing protocol (Convex), the LP/farm protocol (Curve) and the underlying tokens (DAI, USDC, USDT and CRV).
- Portfolio Coverage** - covers a particular wallet up to a specified value of loss, including all applicable assets from insured protocols which the wallet holds. **For example:** [Solace's Portfolio Insurance cover](#).
- Custody/Custodian Coverage** - covers risks associated with custodial wallets and arrangements with centralised entities, including halted withdrawals, custodian hacks and haircuts arising from mismanagement. **For example:** [InsurAce's Custodian Risk Cover](#), and [Nexus's Custody Cover](#).
- De-peg Coverage** - covers risks in an individual token product that arise as a result of the value of the asset de-pegging (i.e. failing to hold market value in line with the underlying assets that it is supposed to reflect). **For example:** [InsurAce's Stablecoin De-Peg Risk Cover](#) and [Nexus's Yield Token Cover](#), which includes a number of specific de-peg risks.
- Staking Coverage** - covers several risks that arise out of blockchain network staking and validating, such as missed rewards and slashing losses. **For example:** [Nexus's ETH2 Staking Cover](#), which specifically covers Ethereum's proof-of-stake beacon chain.
- Bundled Product Coverage** - covers a combination of different protocols, products or different risks to create a single product that covers the users' overarching needs. **For example:** [InsurAce.io's CRV+CVX Bundled Cover](#), which targets Convex users with additional coverage for failures in Curve (which Convex is built on).

## Risks

Though Beefy advocates for the benefits of insurance, we also warn our users that insurance products themselves give rise to a number of risks which may result in your investment in the products being lost. In particular, we'd suggest users consider:

- Coverage Amount** - insurance products typically require that you state upfront the amount of coverage that you want to purchase, typically denominated in fiat currencies. You may be encouraged to buy coverage up to the current value of your investment, or even above that. The coverage amount is typically not a guarantee of the amount you will be paid out, but does limit the maximum value of a pay-out. Furthermore, as the value of your assets change, the suitability of your coverage level will be impacted. For instance a sharp drop in the value of your portfolio will reduce the likelihood of claiming for the full coverage amount you obtained beforehand. Be sure to actively monitor your coverage levels.
- Coverage Limitations** - coverage tends to be constrained to a closed list of claimable risk events (e.g. smart contract vulnerability covers "malfunction or programming flaw, or unauthorized, malicious, criminal attacks, hacks or exploits of any malfunction or programming flaw"). This may mean certain foreseeable events that you want cover for are not included in the product (e.g. a de-peg does not fall within the above definition, so is not covered by smart contract vulnerability cover).
- Captured Products** - insurance products do not always seek to actively point out the limitations of their coverage and which products aren't captured. For instance, portfolio coverage may be tied to products which can be identified by the protocol, and may miss newer products or blockchains, resulting in inadequate coverage for your needs. As the product is general and the UI is user-friendly, you likely won't be told until your claim is rejected that specific products weren't captured in the coverage you purchased.
- Pay Out Process** - all insurance products have strict conditions for paying out for a claim which must be met. For instance, some protocols require members to vote/decide on a claim before the protocol will pay out, which is a factor that is typically outside of users' control and unrelated to the pay out event. Be careful to check the process carefully when assessing whether a product is right for you, including assessing how frequently the product pays out for claims.
  - For example, insurance covering Beefy's protocol or smart contracts will likely pay out if our contracts fail, but will not pay out if the underlying assets that our contracts are built on top of fail. This happened with the Terra UST collapse, where our smart contracts worked as intended, even as the underlying assets' value sank towards zero.
- Providing Coverage** - some decentralised insurance providers (e.g. InsureDAO, Bridge Mutual) offer peer-to-peer insurance products, where users can provide coverage to the protocol as an investment in exchange for a share of the premiums paid by coverage purchasers. Beefy warns our users that these products expose coverage providers to an asymmetric risk of losing your **entire** investment if there is a pay out event under the policy.
  - Providing insurance coverage is effectively a bet on the security of the underlying protocol or smart contracts. Though our SAFU practices are Beefy's top priority, we would not encourage ordinary users to bet on the security of any rapidly changing DeFi technologies.
  - Where Beefy can offer you a similar rate of return on your assets whilst avoiding the pay-out risk, we know where we'd rather invest our funds... 🤖 🤖 🤖

## Partner Products

To bring the benefits of DeFi insurance to our users, Beefy has partnered with a selection of the leading insurance providers in the space, to create products which offer coverage over our protocol, products and contracts. At the time of writing, these are:

### InsurAce.io

InsurAce is a multi-chain insurance provider deployed on a range of chains including Ethereum, BSC, Avalanche, and Polygon. It offers coverage against Beefy smart contract vulnerability over more than half of the chains Beefy is deployed on.

The precise make up of InsurAce's organisational is not entirely clear, though InsurAce has some incorporated legal forms (e.g. InsurAce Global Limited in the UK). It's protocol is made up of two parts: an insurance arm and an investment arm. The investment arm generates profits from fees collected to fund their insurance activities. Users pay a fee (typically 2-5% APY) when using covered smart contracts and are reimbursed for their deposits if it fails.

InsurAce operates a dedicated claims department that will review the different types of claims that can be submitted and vote on the outcome. The claim is first investigated by InsureAce's Advisory Board, which includes experts in Insurance, Security and Legal/Compliance. InsurAce are also aiming to develop a decentralised governance process over time, including the use of community Claim Assessors (who are SINSUR holders that have staked their tokens on the platform) to decide the outcome of claims. However, at the time of writing, the [Claim Assessor page](#) is not live, and the current state of affairs is not made clear on the protocol site.

For more information, see our blog post [here](#) and find the InsurAce Beefy product on the coverage products page [here](#) (Product ID #110). You can find raw data on InsurAce's coverage [here](#).

### Nexus Mutual

Nexus Mutual is an Ethereum-based insurance alternative that provides coverage against specific protocols, yield tokens and custodians (i.e. centralised exchanges). It offers protocol coverage for Beefy across the vast majority of our deployed chains.

Unlike most other providers, Nexus is a discretionary mutual whose members receive insurance-like protection, though it doesn't formally sell insurance. The project uses aligned incentives to enable risk-sharing among all its members, as members act as risk and claim assessors, whilst contributing capital to the protocol. Ownership of Nexus Mutual's token, NXM, is used to buy cover and navigate the claims and risk assessment process. Holders can also take part in the Mutual's governance too.

Nexus offers a range of products including general protocol cover (i.e. covering all different products offered by the protocol) as well as "yield token cover", which covers the full stack of a yield bearing asset (e.g. base assets, liquidity pool and autocompounding vaults). Nexus also offers other novel products such as Ethereum 2.0 staking cover, and custodian cover which protects against the risk of loss caused by keeping assets in the custody of a centralised exchange.

For more information, see our blog post on the partnership [here](#), and the Nexus Beefy product page [here](#). Details of the different types of coverage and their terms and conditions are available in Nexus' documentation [here](#).

### Solace

Solace is a decentralised, DAO-run insurance protocol, on a mission to forge innovation into intuitive protection tools for crypto explorers. It offers coverage for funds invested in Beefy as part of its all-encompassing portfolio insurance product.

By contrast to others, Solace has adopted a novel approach to claims (its optimistic payouts system), wherein users are not required to file claims following an insured event, but are instead automatically paid out where Solace's risk management team assesses that any event has occurred through on-chain analytics. For the future, Solace is also developing a parametric automated claims assessment system, which will be used to automatically quantify loss events using on-chain analytics.

Solace also provides an additional product (Solace Native) for protocols to cover their own smart contracts. This involves protocols holding underwriting tokens, which can be used to vote on a gauge allocation for their protocol among the wider underwriting pool.

For more information, see Solace's twitter thread on the Beefy partnership [here](#), and check out Solace's documentation [here](#).

Please note that Solace's portfolio insurance is tied to their implementation of Zapper's tooling for detecting the value of assets in your wallets. Where these tools do not detect your Beefy assets (e.g. if they're newly issued), you may not be able to recover for those assets under your policy. Always make sure to thoroughly check the "My Portfolio" and "Quote Simulator" parts of your "My Policy" page on Solace's app to check what their Zapper implementation can detect, as this is the limit of what's covered by your policy (even if you purchase more coverage!).

## Comparison

Criteria	InsurAce.io	Nexus Mutual	Solace
Deployed Chains	Ethereum, BSC, Polygon, Avalanche	Ethereum	Ethereum, Polygon, Aurora, Fantom
Beefy Chains Covered	9 chains, including BSC, Polygon and Fantom	15 chains including BSC, Polygon and Fantom	Not stated. Portfolio detected through Zapper. Includes coverage beyond deployed chain.
Product Types	Smart Contract, Custodian, De-Peg, Bundled.	Protocol, Yield Token, Custody, Staking (Smart Contract phased out)	Portfolio (and Native)
Beefy Product Types	Smart Contract	Protocol	Portfolio
Claim Assessment	Private Advisory Board	Public Members Vote	Private Automated Claims

## Unaffiliated Products

As Beefy is entirely open source and public, anyone is free to build on top of our products and code. However, this transparency can cause some confusion for ordinary users as to which products are built and tested by our team, and which aren't. Some insurance providers seek to add to this confusion by stating that their products are "official" or "verified", when they are nothing to do with Beefy. It is ultimately up to the user to do your own research and check that the product is what you think it is.

Below is a list of known products insurance products that offer coverage for Beefy's products or protocol, but are not officially offered in partnership with Beefy and have not been formally verified by our teams. Use at your own risk, and always do your own research.

Provider	Insurance Type	Blockchain(s)	Smart Contract(s)
InsureDAO	Smart Contract	Optimism	0xe3f491c575e02902342ef8488bb3d6c392869fda
Bridge Mutual	Smart Contract	Polygon	0xf0E147862E069460D2ea8837f65aD5D2CaC2D14

Readers are warned to note and consider the "Providing Coverage" risk in the **# Risks** section above when considering these unaffiliated products.

# Team & Goals

:

## When was Beefy launched?

Beefy was born on September 21, 2020, the day our [governance distribution contracts](#) went live. Our first set of vaults opened on October 8, 2020.

## How is Beefy organized?

Beefy is a decentralized working hub for people with a vision to come together and build the future of global finance. Smart contract devs, UI, UX, strategists, statisticians, designers, and artists - anyone can join and contribute (no matter your nationality, sex, or views). By investing in Beefy, you are investing in the idea that a group of highly technical individuals can safely, securely and creatively leapfrog the dinosaurs of traditional finance.

## DeFi, anonymity and Beefy

Personalities get in the way of projects, and we believe Beefy speaks for itself. By having a team that operates anonymously, even amongst itself, we can focus on providing the best experience for our users. That's because we believe the strength of Beefy comes from what we build, which is an opportunity for investors to both automate and maximize the return of investment of their holdings. We urge anyone with concerns that anonymity diminishes credibility to join our Discord community and get a first-hand feeling for the strength and depth of the project.

That being said, Beefy is not fully anonymous. Team members go to conferences and get interviewed, both requiring physical presence. Integrations with large crypto exchanges, like Binance, requires providing personal information.

## How can I get in touch with Beefy?

Our global community managers and team members can be contacted anytime through our official [Telegram](#) and [Discord](#) channels. We should mention that generally speaking within investment communities there are some bad actors around looking to scam, phish, or maliciously target users. Please double check the validity of whoever you talk to, and remember that no one representing Beefy will ever ask you to provide wallet keys or recovery codes.

# Contributor Compensation

Last Update: February 2023

At the heart of Beefy is our culture of innovation and rapid development. From Day 1, Beefy's contributors have worked tirelessly to produce, distribute and iterate on products at lightning speed, both with our own work and in collaborating with the wider DeFi community.

Effective compensation and rewards are essential to building this culture, in order to continue attracting and developing talent and growing the DAO. In the spirit of openness and transparency, this page explains how we at Beefy think about compensation, and our initiatives to motivate engagement and growth.

📌 Note that access to the archived Beefy voting site is available at <https://vote-archive.beefy.finance/>. The archived site requires users to connect their wallet to both the site and the BSC Mainnet to display historic proposals. The current voting site is available live at <https://vote.beefy.finance/#/>.

## Token Distributions

At Beefy, we pride ourselves on our simple but highly effective tokenomics design, which avoids relying on governance tokens as a means of effective compensation. As explained in the [Tokenomics & Governance section](#), our governance token - BIFI - has a fixed supply capped at only 80,000 tokens, so no new tokens can be minted to compensate our members. In fact, over 90% of the total BIFI supply was distributed to the cowmoonity in the first 2 months after Beefy went live in September 2020, so the Beefy [Treasury](#) doesn't contain a stockpile of freshly minted BIFI either.

Rather, we prefer to focus on two use cases for BIFI: governance rights and protocol revenue, paid out through our BIFI Maxi and BIFI Earning Pool vaults. We feel these mechanisms encourage holders to cherish their BIFI and share in the protocol's success as owners, rather than investors or speculators.

The consequences of this design on compensation are clear: our Cowmoonity is encouraged to acquire and hold BIFI for themselves to participate in our success, but we have a strong preference for any payments to be made in stablecoins, rather than BIFI. Aside from the last remaining emissions from the 10% founder's fund (due to expire in July 2022), Beefy does not use BIFI as a form of compensation.

## Retroactive Payments

Given that the BIFI token wasn't designed for payments and compensation, the default method for compensating individual contributions has historically been retroactive payment proposals. Cowmoonity members who have found ways to bring value to Beefy are free to get stuck in and prove themselves useful, and instances of good work (if made known) will be rewarded.

Contributors are able (and encouraged) to submit "Requests for Funds" or "Retroactive Reward" proposals on Beefy's Snapshot page, for deliberation by the Cowmoonity. Proposals can include a range of options for different levels of rewards, or just a simple fixed amount. Proposers are encouraged to give details of the work they've done, or make themselves available for questions in the [#proposal-discussion](#) channel on the Beefy Discord.

In the early months of Beefy's development, retroactive rewards were an effective means of compensating the cowmoonity's efforts. Though our approach to compensation has advanced since then, today they continue on as a useful way to introduce newer or lesser known contributors to the cowmoonity, and compensate individuals for support work that might be difficult to value.

## Developer Incentives

By contrast, product development work (such as building new Beefy vaults) is far easier to value by measuring the profits the products generate. One-time payments are therefore clearly an inferior form of compensation for product development, as we can't know the value of future profits successful products will produce. Similarly, though we believe that holding BIFI over time will also return the output of our developers' hard work to the cowmoonity, we recognise that this does not incentivise developers to participate over and above ordinary BIFI holders.

In recognition of these issues, and the value of incentivising product development, Beefy therefore offers developers a 0.5% share of all harvests for any vaults they successfully develop and deploy (as part of the performance fee structure). Anyone is welcome to try their hand at becoming a Strategist and developing their own vaults using Beefy's open source code, and our [nifty guide](#).

A similar incentives problem also arises for the maintainers of our codebase, as cowmoonity-endorsed compensation will likely always undervalue a critical bug fix (particularly when compared to the reward the fixer could have received by exploiting the bug for theft or ransom). To compensate these contributions, we have also launched an additional [bug bounty program](#) with Immunefi - a crypto bug bounty platform for whitehat hackers. Successful claimants can receive up to \$75,000 for exposing significant exploit, attack or theft opportunities.

## Reoccurring Payments

By August 2021, Beefy's development team had grown to over 20 active contributors covering a range of functions, including smart contracts, UI/frontend, backend/APIs, data engineering, maintainers, DevOps and security. Many of these functions weren't captured by any of the existing developer incentives, and many devs had contributed months of labour without reward or compensation. To address this, Beefy founder Sirbeefalot considered a range of measures to further support and incentivise development and innovation in a sustainable manner.

The first major change was a move to a more holistic rewards framework for the entire dev team. Sirbeefalot devised a new [proposal](#) format in August 2021, which required existing devs to provide details of all of their unpaid contributions in recent months, together with estimates of hours worked and the complexity of their tasks. These notes were published to the DAO, and a budget allocation was proposed, which rewarded each dev with a proportion of the total payment aligning with their individual contributions. On 24 August 2021, 99.82% of the DAO voted to award the highest reward payment of \$150,000 to the dev team, demonstrating the cowmoonity's strong support of the change.

Though grouping developer compensation together was a clear step forward, the process of gathering and analysing unpaid contributions was cumbersome and inconsistent. Following a re-organisation of leadership roles in September 2021, a further [proposal](#) was put forward by Weso in November 2021, proposing to move to reoccurring monthly payments for defined roles across the DAO (both developers and other contributors). In an accompanying forum post, Weso explained the different core contributor roles, and proposed that the DAO would need to regularly re-approve the reoccurring payments. He also suggested that the DAO should retain authority to remove roles and amend or remove proposed payments. On 8 November 2021, the cowmoonity approved Weso's proposal with a consensus of 97.69%.

Since that time, reoccurring contributor pay requests have been repeatedly authorised by DAO votes. The full list to date is:

Period	Amount (\$USD)	Proposal
November 2021	94,000	<a href="#">Archive Page</a>
December 2021 to January 2022	97,500	<a href="#">Archive Page</a>
February 2022 to April 2022	102,000	<a href="#">Snapshot Page</a>
May 2022 to July 2022	127,500	<a href="#">Snapshot Page</a>
August 2022 to October 2022	102,400	<a href="#">Snapshot Page</a>
November 2022 to January 2023	104,000	<a href="#">Snapshot Page</a>
February 2023 to April 2023	104,400	<a href="#">Snapshot Page</a>

## BeefyGrants

The second limb of Sirbeefalot's efforts to incentivise development was the introduction of the BeefyGrants system. In a [paper](#) and blog post introducing the idea, Sirbeefalot explained that the system would be aimed at giving direction to Beefy's development efforts, creating a virtuous cycle of fund allocation, and fairly rewarding community members for their contributions.

Since that time, a range of BeefyGrants and other initiatives have received DAO support and funding, leading to a range of unique and exciting Cowmoonity-led initiatives. As such, our contributors are encouraged to consider applying for larger-scale grants and funding arrangements to support their ideas for new products and initiative.

# Governance

Last Update: December 2022

Beefy is run as a decentralised autonomous organisation (DAO), meaning it is controlled and directed by a decentralised network of contributors and community members, who enable Beefy to function on its own. This page outlines how the governance process behind our DAO works, and how you can get involved.

## How is Beefy governed?

As a decentralised organisation, Beefy is proudly governed by our \$BIFI token holders. This includes our founders, Core contributors and the wider Cowmoonity, as well as some external holders. Most major decisions that Beefy takes are put to and voted on by our \$BIFI holders, including how our fees are set, how the protocol and its contributors are funded, how we market Beefy and what direction we should take on certain decisions. Think of our BIFI holders as members of the Beefy legislature.

The key mechanism for this is governance voting, which is undertaken through our [Snapshot](#) page. BIFI holders are entitled to raise proposals (if they hold at least 1 \$BIFI) and vote on them. Discussion of all proposals is actively encouraged on our social media channels. In particular, we would encourage you to head to the [#beefy-proposals](#) and [#beefy-proposal-discussion](#) channels on the [Beefy Discord](#) server. In the threads for the [#beefy-proposals](#) channel, you'll find a dedicated space for discussion of individual proposals, or [#beefy-proposal-discussion](#) serves as a more general forum.

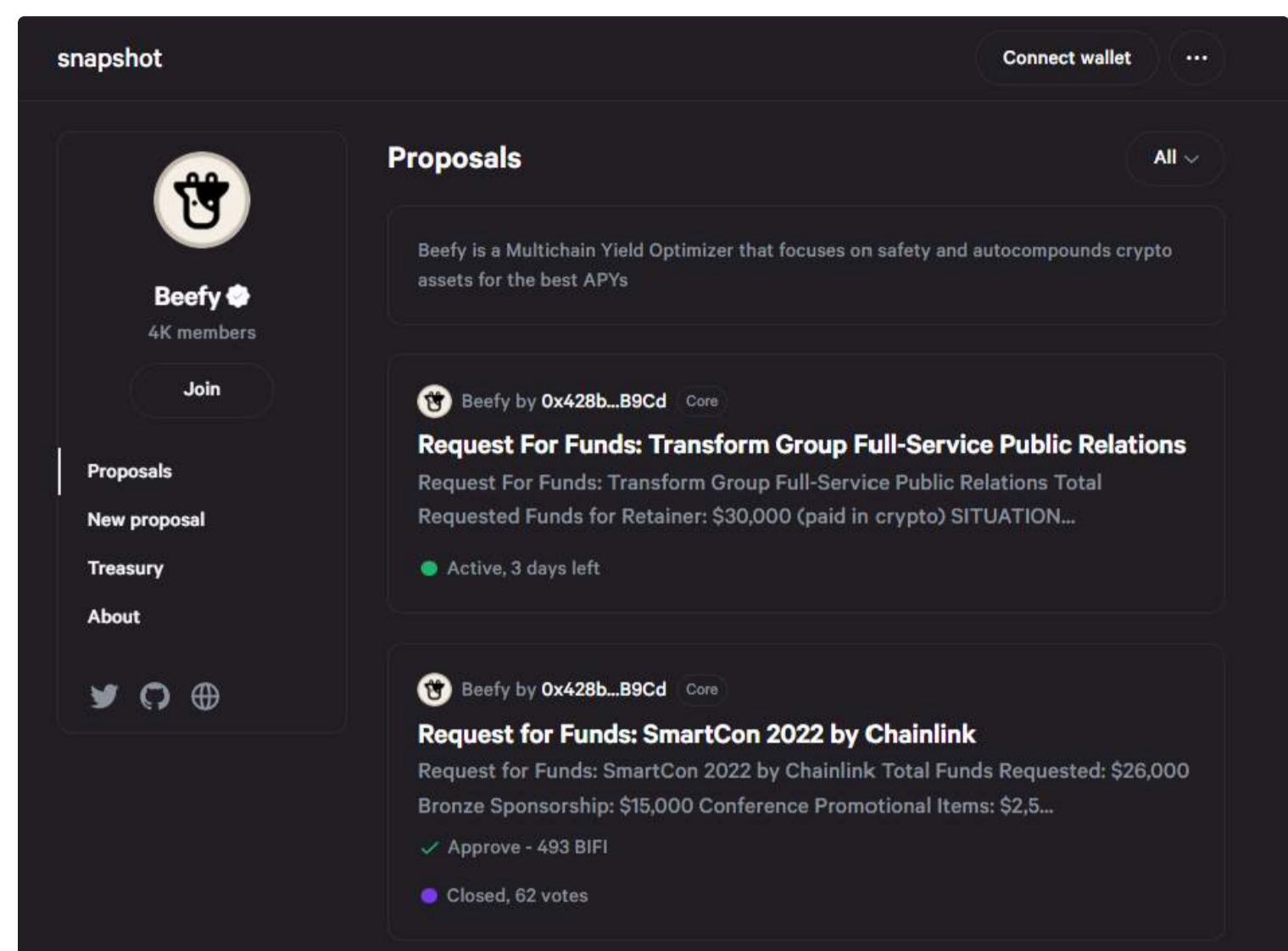
In addition, the day-to-day operations of Beefy are governed by our Core contributor team, who take the role of the executive wing, and are tasked with carrying out the will of our BIFI holders. As with any growing organisation, it's impossible to run every decision through formal governance, so our Core team have been delegated the necessary authority to look after the protocol. That said, any area of Core's decision-making can instead be raised through governance voting, to ensure that they may be held to account.

## How do I take part in governance?

By simply holding \$BIFI, even if staked in the native token Earnings Pool or BIFI Maxi vault, a user earns the right to create proposals and vote in them. Voting sway and power are derived from the \$BIFI holdings of the participant. The reasoning behind this follows that those holding more \$BIFI are more invested in the project, and therefore have a larger incentive for the platform itself to succeed and prosper.

Beyond voting for yourself, you can also register to participate as a delegate, so others can delegate their voting power to you for inclusion with your own vote. Through this mechanism, trusted voices in the community can leverage their support with a small amount of effort on the part of their supporters. It also allows those short on time to ensure that their \$BIFI is participating in governance, without requiring them to engage with every proposal. See below and the [DelegateRegistry Contract](#) page for further details.

You can see proposals and vote on them yourself by heading to [vote.beefy.finance](https://vote.beefy.finance).

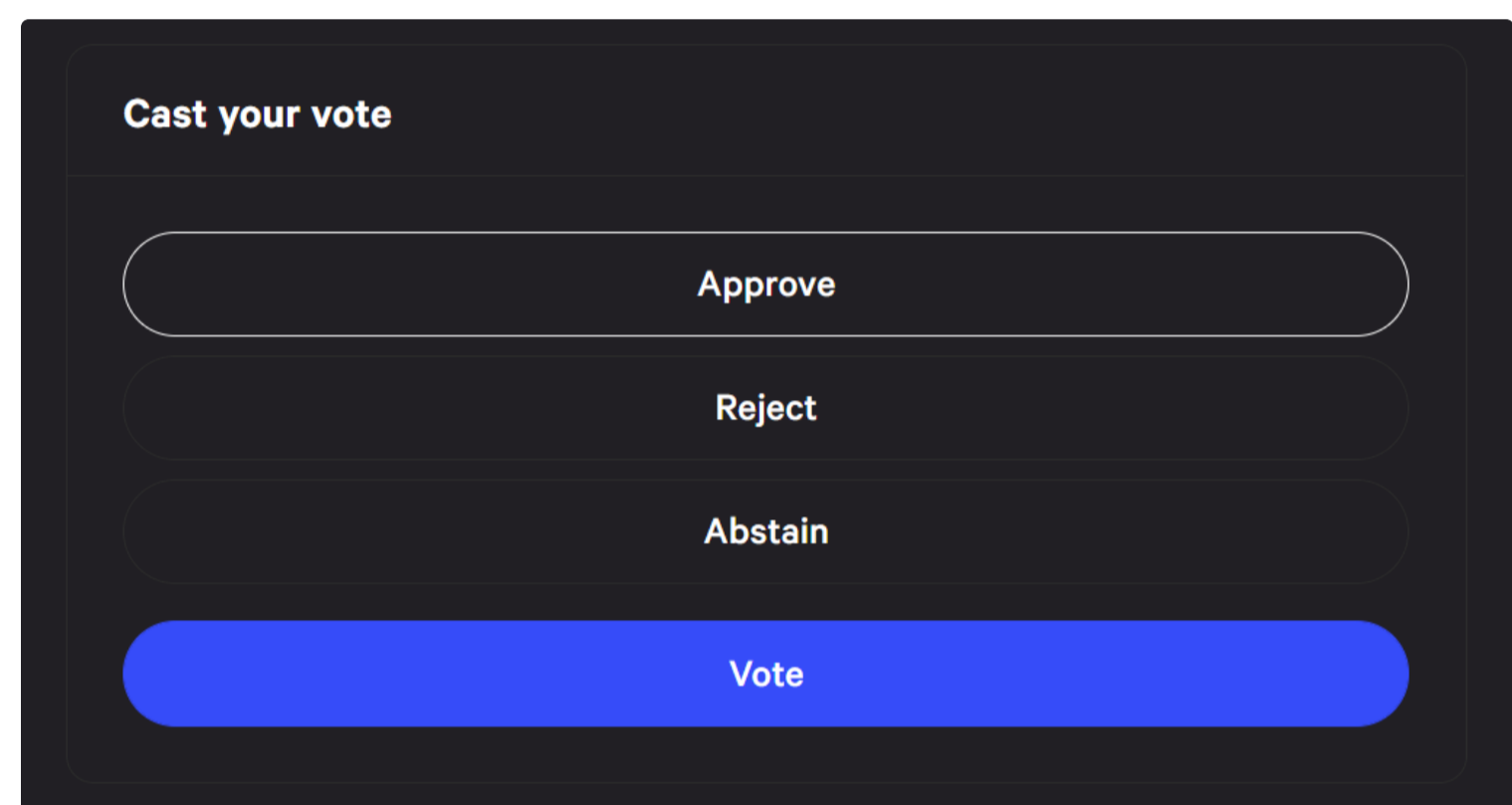


Beefy's Snapshot page houses our governance proposals and voting.

## How do I vote?

Voting requires you to hold \$BIFI, which can either simply be held in your wallet or staked in the native token Earnings Pool or BIFI Maxi vaults. You do not need to remove your stake either to vote. Voting power is based directly on the amount of \$BIFI each voter holds.

To submit your vote, simply head to our [Snapshot](#) page, connect your wallet to Snapshot and then head to an open proposal you wish to vote on. Here you'll find an interface to "Cast your vote" by selecting your preferred option(s) and clicking "Vote". You'll then be required to sign a transaction through your wallet to formally submit your vote.



Look for the "Cast your vote" box on an open proposal page to get involved.

## How do I delegate my vote?

You can delegate your vote on BNB chain by interacting directly with the [DelegateRegistry](#) contract. A full tutorial of how to delegate is available in the [DelegateRegistry Contract](#) page at [#Delegation Walkthrough](#). Please note that, due to our bespoke Snapshot tooling, you must delegate on BNB chain for your delegation to be effective. Delegations on other chains may register in the relevant DelegateRegistry contract, but will not be recognised by our tooling.

## How do I become a delegate?

If you'd like to represent the Cowmoonity by acting as a delegate for others to direct their voting power to, you can do so by reaching out to us and providing your name and public wallet address. A full list of the current delegates offering to represent your interests is maintained in this [Google sheet](#). We'd recommend that budding delegates seek to provide as much information as they can for the sheet, for the benefit of users who may delegate to them.

## How are votes counted?

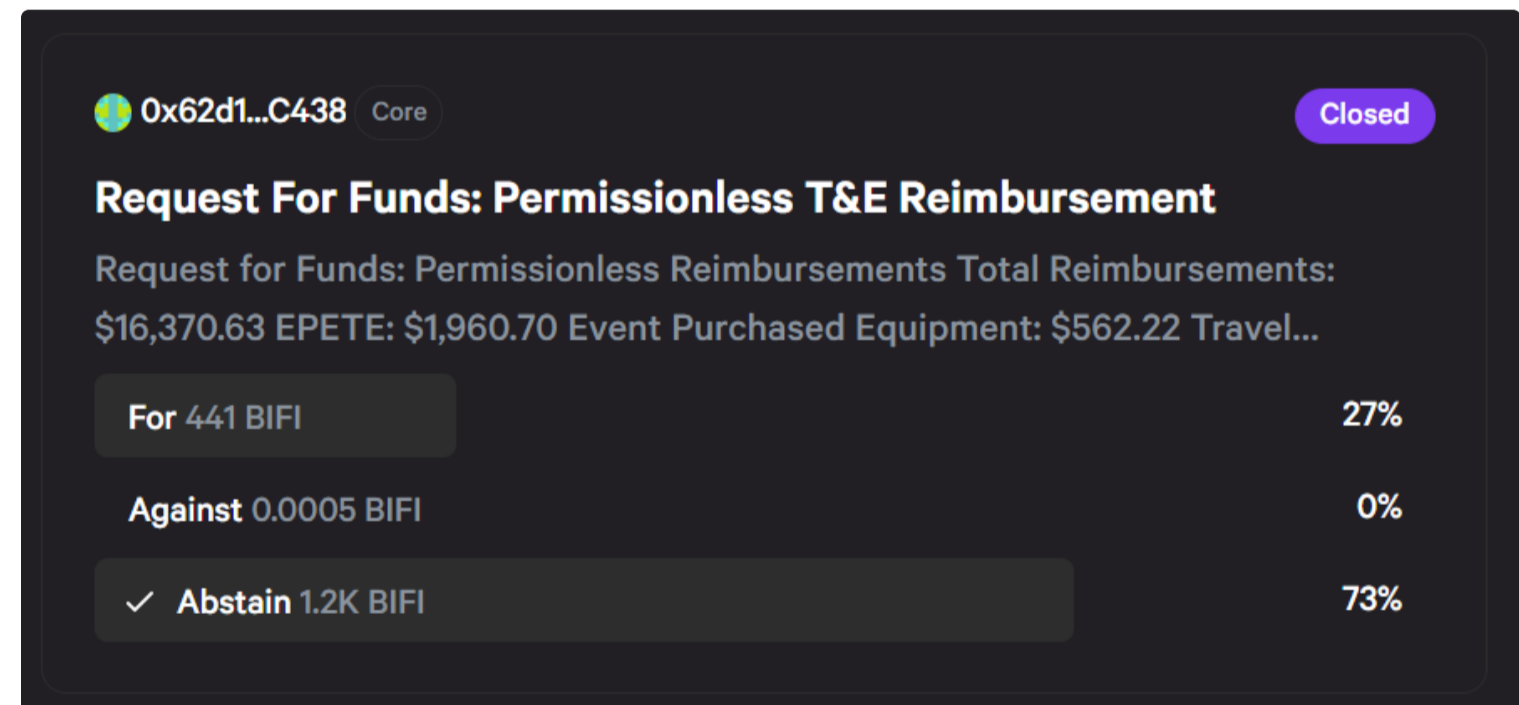
For each proposal, a snapshot of the blockchain is taken at the time of posting and used to calculate how many \$BIFI tokens are held by each voter. This voting power is locked from the time of posting, so you can't buy more \$BIFI to influence a live vote.

Through our Snapshot page, proposals can take a number of different forms, each of which weigh votes slightly differently. The different options include:

- Basic voting - voters can either approve, reject or abstain from the vote, with the majority (absent abstentions) winning the vote.
- Single choice voting - voters can choose one of several choices, with the option with the largest proportion of votes winning.
- Weighted voting - voters can spread their voting power across multiple choices, with the option with the largest proportion of voting power winning.
- Quadratic voting - weighted voting, but with voting power adjusted logarithmically, so that smaller voters have more voting power per token than larger voters. The option with the largest proportion of voting power wins.
- Ranked choice voting - voters can select several options by an order of preference, with votes then counted by eliminating the option with the least highest preference votes, and reallocating those votes to their next highest preference, until only one option remains.

**ⓘ** Please note that for all Beefy governance votes which include an "abstain" option, a vote to abstain will be interpreted as a vote for the side that has the most votes, when ignoring the abstaining votes. If a quorum is introduced and is met only through the inclusion of abstaining votes, this remains a valid proposal, and the outcome will be accepted.

The Snapshot user interface may recognise the outcome of a vote as "abstain" where abstaining votes are the largest group, though the outcome will in fact be the option with the most votes otherwise. See for example the outcome of the [Permissionless 2022 reimbursement](#) proposal below.



Where abstain receives the most votes, the outcome with the second most will be adopted.

## What is the quorum for voting?

To ensure that proposals that don't receive much interest aren't passed by default, Snapshot includes an option for a voting quorum, where a proposal needs to receive more than a specified proportion of total available votes in any direction to be considered valid.

No formal quorum has ever been adopted by Beefy, so no quorum has applied to any of our previous governance proposals. However, in some historic proposals, our Snapshot space has included references to the Snapshot quorum in the default user interface, even though these did not in principle apply to the formal proposal. Please note that where any such historic proposal appears to have failed to meet the stated quorum, it is still considered to have been adopted by the DAO.

## How do I create a proposal?

Proposals can be created if the submitter holds or stakes at least 1 \$BIFI. Simply visit <https://vote.beefy.finance/#/> and click New Proposal.

Each proposal is made up of a question to pose to the community, along with the option of choices that others can vote on in response to your question. Simply set a start and end date for your proposal and publish it.

## What types of proposals are there?

As our governance process has developed, we have adopted a number of different forms of proposal for different purposes. These include:

- Beefy Improvement Proposals (BIPs) - the default form of proposal, for an issue aimed at improving the protocol in some way.
  - This now includes proposals for [recurring contributor payments](#).
- Beefy Signally Proposals (BSPs) - an alternative, non-binding proposal, aimed at establishing consensus amongst \$BIFI holders, or demonstrating their opinions/preferences.
- [Requests for Funds / Retroactive Payments](#) - a simple request for funding from the Beefy [Treasury](#) for a particular work stream or contributors. These are typically a form of expense, rather than an investment.
- [Grant Proposals](#) - a more detailed request for funding from the [Treasury](#), typically relating to a large project that requires significant funding and input, but is aimed at delivering future revenues or cost-savings to Beefy (so is an investment, rather than an expense).

## How do we protect against malicious votes?

It's no secret that open governance brings a range of risks, including malicious votes aimed at hacking the protocol or damaging our Cowmoonity. To protect against this possibility, our Core team actively moderates the governance process, looking to take down any potentially harmful proposals.

This includes proposals that are: (1) clearly harmful or malicious; (2) unfair, impossible or fundamentally irrational; (3) don't allow sufficient time to vote; or (4) which lack sufficient clarity or comprehensibility to be executed. It may also include attempts to force through a proposal by changing the voting system or parameters for a failed vote (e.g. moving to quadratic voting to circumvent large \$BIFI holders), or leaving too little time for voting.

## Where can I find details of past votes?

Recent votes are stored on the our [Snapshot](#) page. At the end of 2021, Beefy transitioned from a custom Snapshot fork running exclusively on Binance Smart Chain (BSC) to a modern instance of Snapshot, capable of supporting each of Beefy's chains. As a result of the transition, proposals ending before the start of 2022 were not carried forward to the new site. Instead, an archived copy of the original site has been preserved.

**ⓘ** Note that access to the archived Beefy voting site is available at <https://vote-archive.beefy.finance/>. The archived site requires users to connect their wallet to both the site and the BSC Mainnet to display historic proposals. The current voting site is available live at <https://vote.beefy.finance/#/>.

Alternatively, we have prepared and maintain a [proposal repository](#) here in our docs, where you can quickly get access to any of the major proposals on either voting site throughout our history.

# Proposal Repository

Last Update: February 2023

Beefy is governed by and for our Cowmoonity through our governance process. Any user can submit a Beefy Improvement Proposal (BIP), Beefy Signalling Proposal (BSP), request for funds or other general proposal for consideration and voting by the Cowmoonity.

This page lists some of the most important proposals from our history, together with links to the full proposal text and results. The list focuses on key initiatives, governance changes, product developments and events in our history. Common matters like individual payments, bounties and partnerships are generally excluded. The page also includes a list of significant proposals that did not receive approval, to inform those considering raising similar issues in the future.

At the end of 2021, Beefy transitioned from a custom Snapshot fork running exclusively on Binance Smart Chain (BSC) to a modern instance of Snapshot, capable of supporting each of Beefy's chains. As a result of the transition, proposals ending before the start of 2022 were not carried forward to the new site. Instead, an archived copy of the original site has been preserved. This page consolidates the two sites, and provide a seamless history of governance at Beefy.

**Note that access to the archived Beefy voting site is available at <https://vote-archive.beefy.finance/>. The archived site requires users to connect their wallet to both the site and the BSC Mainnet to display historic proposals. The current voting site is available live at <https://vote.beefy.finance/#/>.**

## Approved

Proposal	End Date	Proposer
Recurring Monthly Budget for Bribes	14/02/2023	Core
Premium Placement on Coinbase	07/02/2023	Core
[BIP:61] Reoccurring Contributor Pay	24/01/2023	Core
Convention Participation Budget	21/01/2023	Core
Development of New Vault Type	08/11/2022	-
[BIP:51] Reoccurring Contributor Pay	26/10/2022	Core
[BIP:46] Reoccurring Contributor Pay	30/07/2022	Core
[BIP:45] Protocol Sustainability	25/07/2022	Core
[BIP:36] Contract Library and Testing Suite Bounty	13/05/2022	Core
[BIP:35] Reoccurring Contributor Pay	02/05/2022	Core
[BIP:29] Messari Research Partnership	05/04/2022	Core
Invest in MooWars	07/04/2022	Moondance DAO
[BIP:26] ReflectiveChimp into Beefy Core	31/03/2022	Core
[BIP:21] Beefy Product Liquidity	08/03/2022	Core
[BIP:20] AllTrades into Beefy Core	03/03/2022	Core
[BIP:15] Beefy Solidity NFT Decision	16/02/2022	Core
[BIP:14] Beefy Data Project - Infrastructure	04/02/2022	Core
[BIP:11] Reoccurring Contributor Pay	29/01/2022	Core
[BIP:10A] Fantom Validator Platform	28/01/2022	Core
[BIP:10] Fantom Validator Capital	26/01/2022	Core
Market Maker - System 9	26/01/2022	Core
V2 App Loading & Redux Rework	21/01/2022	Core
Marketing Initiative	18/01/2022	Core
Legal Opinion	15/01/2022	Core
Discretionary Funds for Beefy Core	01/01/2022	Core
Reoccurring Contributor Pay	23/12/2021	Core
Beefy/MoonPots Christmas Surprise	21/12/2021	Core
Beefy Avatars - Secondary Markets	19/12/2021	-
Beefy Avatars - Splitting Mint Funds	19/12/2021	-
Beefy Avatars - Funding Request	16/12/2021	-
Beefy Avatars - Mint Price	16/12/2021	-
Update Governance Snapshot Bounty	10/12/2021	-
Wizard-Themed DAO on Beefy	28/11/2021	-
Mai Boost/mooBIFI Collateral Funding	21/11/2021	Core
Beefy Gear V2	18/11/2021	-
Beefy Gear	09/11/2021	-
Play2Earn Game Initiative	09/11/2021	-
Reoccurring Contributor Pay	08/11/2021	Core
Request to Rebalance Treasury	08/11/2021	Core
Decentralized Auto-Harvesting Funding	16/10/2021	-
Beefy V2 Project Management Budget	06/10/2021	-
Thank Moonpot for Successful Launch	03/10/2021	Miyazaki
Buy Beefy.com	03/10/2021	Miyazaki
Mass Quickswap Vault Upgrades	30/09/2021	-
Arbitrum Harvester	30/09/2021	Core
mooBIFI Lending	30/09/2021	-
Beefy Avatars - Funding	28/09/2021	-
Dedicated Beefy Data Bot	20/09/2021	-
BeefyGrants Proposal	20/09/2021	Core
Creation of the Treasury Council	05/09/2021	Core
Retroactive Contributor Pay	24/08/2021	Core
Stablecoin Zapping for All Vaults	06/07/2021	-
Moonpot Initiative Budget	16/05/2021	Core
Convert 50% of wBNB Holdings to Stables	08/05/2021	Core
Form Council of SAFU	04/05/2021	-
Cowmoonity Coordinator Budget Request	23/04/2021	Core
Zapper Tool Structure	21/04/2021	-
Launchpool Initiative Budget Request	13/04/2021	-
Github Issues Bounties	27/03/2021	Core
Publish Our Product Roadmap	16/03/2021	-
Beefy Launchpool Budget Request	06/03/2021	Core
Positioning Copywriting & Marketing Budget	06/03/2021	Core
Beefy V2 UI Budget	03/03/2021	Core
Scrap Harvester Rewards	02/03/2021	-
Generic Integration Tests Bounty	28/02/2021	-
Reusable Vault Upgrade Tests Bounty	27/02/2021	-
Professional Marketing Team for Beefy	24/02/2021	-
Put the Treasury to Work 2	18/02/2021	Core
Reoccurring Fixed Expenses Budget	16/02/2021	Core
Beefy API Pull Requests Budget	12/02/2021	Core
Certik Audit Reservation Deposit	09/02/2021	-
Make Beefy Docs Repository Public	30/01/2021	-
Allocation for Vault Event Monitoring	20/01/2021	-
Stake by Beefy Proposal	20/01/2021	-
SBETH Tutorial Campaign	17/01/2021	-
Social Media Development Tools	06/01/2021	-
Retroactive Contributor Pay	16/12/2020	Core
Make Power Strategy Architect on Discord	27/11/2020	Core
First Public Proposal - Twitter Giveaway	21/11/2020	Core
Prefix for Beefy Vault-issued Tokens	03/10/2020	Core
First Test Proposal - Buy The Dip?	02/10/2020	Core

## Not Approved

Text	End Date	Proposer
[BIP:58] Adopt Governance Guidelines	02/12/2022	jackgale.eth
Upgrade BIFI Maxi/Earnings Pool	02/12/2022	-
Beefy Cow Club	10/11/2022	Tilty, Pepecow and Alderian
Increase Initiative Transparency	30/10/2022	-
Activate beFTM Partial Redemption	16/10/2022	-
Implement New Farms and Token Burns	31/08/2022	-
Renew Unchained Sponsorship	03/06/2022	Core
Launch Incentives for Liquidity Provision	24/05/2022	-
Launch Support for Optimism Blockchain	10/05/2022	-
Beefy.fi Domain Acquisition	24/04/2022	-
MooStarter: Beefy Launchpad	17/04/2022	-
Purchasing Beefy.eth	13/04/2022	alexkagan.eth
Divide BIFI Maxi rewards between blockchains equally	12/03/2022	-
Ukraine Support Banner	28/02/2022	-
Solana Integration Poll	25/02/2022	-
Reinstate Discord Chat EXP in Degen Discussions Channel	11/02/2022	serenity.our-future.eth
Introduce Small Tax on BIFI Transactions	06/02/2022	-
Launch Support for Ethereum Blockchain	31/01/2022	ags.eth
Expand into SmartBCH	31/01/2022	-
Should Beefy Expand their Services?	31/01/2022	-
Beefy Analytics Proposal	30/12/2021	-
Beefy Goes Green! (Fixed Typo)	26/11/2021	-
Launch a Vault on Ethereum	17/05/2021	-
BIFI Holders Head Start on Boosts	08/04/2021	-
10% of Boost Vaults Paid to BIFI Maxi	29/03/2021	-
Start the Cowcommunity/Moo-Cult	24/03/2021	-
More Benefits for BIFI Holders	08/03/2021	-
Project Timelock Monitoring	07/03/2021	-
Use Incognito Chain for Privacy	02/03/2021	-
Launch New Token to Promote Staking	26/02/2021	-
Create Insurance Fund	23/02/2021	-
Move from Discord to Privacy Platform	31/01/2021	-
BIFI Maxi Entrance Fees	15/01/2021	-
BIFI Maxi Airdrop	30/12/2020	-
Put the Treasury to Work 1	12/12/2020	-

## Signalling Proposals

Beefy Signalling Proposals (BSPs) are non-binding proposals intended to signal the Cowmoonity's feelings and intentions on the issue being raised, without tying the DAO to the outcome. This allows moonies to express their views honestly, without concern for tactical voting or the practicalities of implementation.

In theory, a clear outcome in favour of change should be a strong signal to the Core to pursue the change independently. Where the Core refuses or fails to deliver on that signal, a binding Beefy Improvement Proposal can be used instead. Alternatively, a strong signal against proposed changes should quieten pressure for those changes amongst the Cowmoonity, at least for the near future.

Proposal	End Date	Proposer
[BSP:02] Profit Mandate	20/02/2023	jackgale.eth
[BSP:01] Changing Reoccurring Comp	28/05/2022	jackgale.eth



# Treasury

Last Update: March 2023

At the heart of Beefy's operations is our Treasury, which is managed by our Core contributor team. As our \$BIFI token is fully distributed, with no tokens reserved for later use, our entire operation is funded by our ongoing revenue-generating activities. To keep the lights on and our vaults autocompounding, Beefy relies on the constant vigilance of its treasury team to manage the use of our Treasury responsibly, and with the DAO's best interests at heart.

This page explains the history and structure of the Beefy Treasury, summarises our approach to treasury management, and provides the details you'll need to access our live treasury activities.

## History

Beefy's core product - our Beefy Vaults - are designed from the ground up to generate fees for the project (see [📄 Beefy Fees Breakdown](#)) to sustain our activities. As such, we have always been a revenue-generating project, and have always had some level of treasury inflows and outflows.

Over time, our treasury activities have become increasingly sophisticated. Initially, the project's founders privately managed the inflows from the smart contract to cover their costs in operating and maintaining the protocol. The original [BeefyTreasury contract](#) was deployed on 6 November 2020. The Treasury contract was very basic in functionality, allowing the native chain token and ERC-20/BEP-20 tokens to be withdrawn by the owner, which was the [BeefyDeployer EOA](#).

By March 2021, Beefy had expanded to include more contributors, and so the Core contributor team created the [#👁️-treasury](#) channel in the [Beefy Discord](#) as a place for discussion of all treasury-related issues. Over the next few months, Beefy became increasingly multichain, and additional BeefyTreasury contracts were deployed on [Avalanche](#), [Polygon](#), [Fantom](#) together with some other early chains. However, the Core team recognised the limitation of the early Treasury's design, and a move MultiSig wallets was debated, despite MultiSig solutions not being available on all chains.

By September 2021, the DAO had settled on moving to MultiSig wallets on BSC and Polygon (as the two chains with MultiSig solutions available at the time), and establishing the Treasury Council as the signers of the MultiSig. On 3 September 2021, a soft vote was concluded on Discord to appoint the new members of the Treasury Council. This was followed up by a [Snapshot Proposal](#), which confirmed the initial 7 members. On 11 September, ownership of the original BeefyTreasury contract on BSC was transferred to the new MultiSig. Since then, the MultiSig system and Treasury Council has continued and expanded to each of Beefy's new chains.

Also in September 2021, Beefy's Core team began to implement bots in the [#👁️-treasury](#) channel, initially displaying the value of the Treasury for the benefit of DAO members. By October 2022, this had evolved to include each new transactions live when posted from any of the Treasury's multisig wallets. Some of the key bot commands are:

- **Show Full Treasury:** `@Bitch I'm a Cow#9189 treasury`
- **Show Full Treasury on Specified Chain:** `@Bitch I'm a Cow#9189 treasury {blockchain}`
- **Show Current Cowlector Balances:** `@Bitch I'm a Cow#9189 cowllector balances`

## Treasury Council

As described above, the Treasury Council was formed in September 2021 as the team that should conduct and manage all Treasury transactions and related activities for Treasury management. The role of the Treasury Council has expanded gradually, to now include:

- Orchestrating, signing and executing all Treasury transactions (i.e. payments out of Treasury);
- Managing Treasury assets, such as creating and exiting protocol-owned liquidity and holding grants, pending rewards and other external assets;
- Monitoring and providing operating capital (e.g. gas) for Beefy's infrastructure, including our Cowlector, Fee Batch Harvester and contributor EOAs;
- Arranging for payment of key expenses, including contributor salaries, sponsorships, recurring hosting and service fees and reimbursement for contributor expenses; and
- Handling regular bribe payments to partner exchanges.

The Treasury Council always has 7 members, and requires consensus from any 4 to sign off on any MultiSig transaction before it can be executed. Details of the current 7 members and their Treasury EOA wallets are:

- **AllTrades:** `0xa75209dC118dF7B6541db5b7Be0DE9485Ebaa907`
- **DefiDebauchery:** `0x037465bF6a4A8D7F552AE18046478C6A727178F3`
- **Mjoaris:** `0xBFEb0756f09f73A19CE62Fba6a8Db4e922E73A14`
- **Pablo:** `0xDB583b636f995eF1EF28ac96B9bA235916bd1583`
- **Power:** `0x6fCE222540015290FB572C82622dc73a431CdF3F`
- **TBC:** `0x428b2F01Bfb0917FE6FF463f37B0c47F1782B9Cd`
- **YR2150:** `0xF24f55d6765D559BFF4C5557dD9024CBA10d30e`

Any questions for the Treasury Council should be directed to the [#👁️-treasury](#) channel on the [Beefy Discord](#). For security purposes, please do not directly message Council members instead, or your messages will be ignored and potentially blocked or reported.

## Treasury Infrastructure

Beefy's Treasury relies on a few core building blocks of infrastructure to facilitate the secure inflows and outflows of funds from Treasury. The core elements are:

- **Treasury MultiSig** - multi-signature wallets controlled by the Treasury Council, used to hold all of the main assets of the treasury securely, and in a manner that can't be exploited by any single individual;
- **Treasury Contributor Wallets** - externally-owned accounts (EOAs) controlled by different Beefy contributors, such as our Treasury Council members and other key DAO contributors. By using EOAs to interact with external protocols and contracts, we isolate the risk of hacks or exploits impacting the Beefy Treasury, by isolating it from externals and giving no or very limited external approvals;
- **Cowlector** - the Beefy Cowlector is a bot created and maintained by the Beefy Core which seeks to harvest Beefy vaults regularly wherever a profitable opportunity presents itself. As the harvest caller claims a small fee as a proportion of the harvest, the Cowlector analyses the size of the potential reward versus the cost of harvesting, and makes automatic calls where appropriate; and
- **Fee Batcher** - the Beefy Fee Batcher is another bot created and maintained by the Beefy Core team which aggregates Beefy's fees from vault harvests in one place, to be able to efficiently swap aggregated batches of fees into the main currency of the chain, before sending the output of the fees in one neat batch to the Beefy treasury.

The Treasury MultiSigs can be found at the following addresses for the following chains:

- Arbitrum: `0x3f5eddad52C665A4AA011cd11A21E1d5107d7862`
- Aurora: `0x088C70Ddff3a3774825dd5e5EaDB356404248d83`
- Avalanche: `0x26dE4EBffBE8d3d632A292e972E3594Fc2eCeEd`
- BSC: `0x7C780b8A63eE9B7d0F985E8a922Be38a1F7B2141`
- Canto: `0xF09d213EE8a8B159C884b276b86E08E26B3bf75`
- Celso: `0xca807d809f9639cefb3d31a7951ce3ab405a2fd`
- Cronos: `0xa9721Ae5042482D7a884A2138f580459B680920f`
- Ethereum: `0xc9C61194682a3A5f56BF9Cd5B59EE63028aB6041`
- Emerald: `0x8fd0869271d26ee6653f5d5650685630f75b6aedf`
- Fantom: `0xdFf234670038dEFB2115Cf103F86dA5fB7CfD2D2`
- Fuse: `0x1C124c2CaB83b3C3B5D0f0899CeeA5e06964599F`
- Harmony: `0x523154a03180FD1CB26F39087441c9F91BcD0389`
- HECO : `0xdbb72c8b7ebdd52a4813b9d262386dfab69c9ba`
- Kava: `0x07F29FE11FbC17876D9376E3CD6F2112e81feA6F`
- Metis: `0x0f9602B7E7146a9BaE16dB948281BebDb7C2D095`
- Moonbeam: `0x3E7F60B442CEAE0FE5e48e07EB85Cfb1Ed60e81A`
- Moonriver: `0x617f12E04097F16e7393484f35175a1B8196551`
- Optimism: `0x4ABa01FB8E1f6BF80c56Deb367f19F35Df0f4aE`
- Polygon: `0xe37dD9A535c1D3c9fC33e3295B7e08bD1C42218D`

## Treasury Management

Management of Beefy's Treasury has also evolved over time, to reflect the changing needs of the protocol and the DAO. Operating on so many chains, each with their own sets of vaults, infrastructure and MultiSig wallets makes it a complicated process to actively manage the Treasury. Instead, a number of overarching principles have emerged to guide current best practices.

First and foremost, Beefy's contributor team have opted to operate the Treasury on each major chain primarily in stablecoins, to mitigate the downside risk of holding volatile assets. This allows Beefy to make clear and deliberate decisions about how to use its funds, free from the risks of current market condition. As such, the Fee Batcher on each major chain swaps all Beefy's vault fees into a designated currency for that chain, as detailed in [# Inflow Currencies](#) below.

Secondly, it was decided that investing a portion of treasury assets in Beefy Vaults to generate passive income would be a sensible and smart use of funds. Generally speaking, only a small proportion of the Treasury held on our major chains is ever invested in Beefy Vaults, and they are always allocated to stablecoin-only vaults. Though returns on invested funds are generally relatively small, this mechanism does help to mitigate the inflationary nature of stablecoin valuations.

Finally, to facilitate liquidity for our tokens across the different chains, Beefy also invests in and maintains liquidity positions in our tokens, including both our [👁️ \\$BIFI Token](#), and our [👁️ Beefy-escrowed Tokens](#). Our \$BIFI token is typically paired with native or gas tokens on the relevant chain, to provide the easiest route into the token for new arrivals on the chain. Once the bridged \$BIFI token contract is deployed on a new chain, the contract is connected to the Beefy Bridge and our partners at Multichain, so that more \$BIFI can be bridged onto the chain and so that an initial LP can be created.

Live summary details of the current state of the Beefy Treasury, including all liquid, staked/invested and locked tokens and liquidity positions, are available in the [Treasury Dashboard](#) page of the Beefy App.



The Beefy Treasury Dashboard was introduced in January 2023 to provide immediate summary insights into the current state of the Beefy Treasury.

## Inflow Currencies

As described above in [# Treasury Management](#), inflows on our key blockchains are swapped into stablecoins for treasury management purposes, with each chain's Fee Batcher adopting a key stablecoin for its operations. The following chains have the following nominated currencies:

- Arbitrum: USDC
- Avalanche: USDT
- BSC: BUSD
- Canto: USDC
- Cronos: USDC
- Ethereum: USDC
- Ethereum Validator: ETH (retained in validator)
- Fantom: USDC
- Fantom Validator: WBTC (gradually bridged to Ethereum)
- Fuse: BUSD
- Fuse Validator: Fuse (retained in validator)
- Kava: USDC
- Metis: USDC
- Moonbeam: MAI
- Moonriver: USDC
- Optimism: USDC
- Polygon: USDC

Volumes and liquidity on Aurora, Celso, Emerald, Harmony and HECO are all currently too small to warrant stablecoin management, so vault fee inflows are typically swapped into the relevant native chain token instead.

# Cowmoonity

Last Update: February 2023

## Core contributors

Beefy's Core contributors are working hard every day to improve Beefy in various domains. Most have a paid function.

Name	Discord	Twitter
chebiN	chebiN#2624	-
DefiDebauchery	DefiDebauchery#3115	@DefiDebauchery
Dieter	Dieter#8010	@Dieter_Design
EPETE	EPETE#3333	@_ePete
Eren	Eren Jaegar#9562	@eren_beefy
frondoto	frondoto#1954	@frondoto1
jackgale.eth	jackgale.eth	@iamjackgale
kexley	kexley#7853	@kexleyBeefy
mjoaris	mjoaris#5328	@mjoaris
MrTitoune	mrtitoune	@ClemToune
Pablo	Pablo#9688	@pablo_beefy
Power - Beefy.Finance	Power - Beefy.Finance#3933	@PowerBeefy
ReflectiveChimp	ReflectiveChimp#4499	@ReflectiveChimp
Roman Monk	Roman Monk#7468	@roman_mnk
TheBeefyCow	TheBeefyCow#9890	@thebeefycow
w18o - BIFI 18 Chains.bifi	Weso#1643	@w3soBeefy
YR2150   Beefy Pulse	YR2150   Beefy Pulse#5907	@yr2150T / @beefypulse
Zapmore	Zapmore#3433	@Zapmore3


## Beefy OG's

Below is the list of Beefy OG's who were around in Beefy's first year, some even from the start, and helped to grow Beefy in the early stages. The stats are based on [Beefy's Discord](#), plus Twitter account if available. This role is no longer given to new contributors. Most of Beefy's Core contributors are Beefy OG's as well.

Name	Discord	Twitter	Primary Role
Oxbeefy	Oxbeefy#0679	-	-
AllTradez	alltradez	@AllTradez	Dev
zxcvzxcv	bagholder#0845	-	-
barnyard	barnyard#6764	-	-
Broken Robot (they/them)	brknrobot#0952	-	Dev
Cassandra	Cassandra#2104	-	-
CHIP - BLUE CHIP/LOW RISK	Chip#1528	@Chip_Penguin	Mod
Destructible Fruit	Destructible Fruit#4578	-	-
Dibby	Dibby#5165	-	-
dingbat	dingbatx#6073	-	Dev
DoD4uFN   BIFI Maxi	DoD4uFN#9980	@DoD4uFN	Dev
dydymoon.eth	dydymoon.eth#3133	@dydymoon1	-
FEEDZED	FEEDZED#9625	-	-
felipe_bifi	felipe_bifi#1224	-	-
GPMS	GPMS#0844	-	Mod
kaltoro	kaltoro#2974	-	-
keyboard	keyboard#3813	-	-
Mo0o0	Mo0o0#1628	-	Dev
mooncow	mooncow#8411	-	-
Moonster	Moonster#4130	@Moonster_BSC	Dev
Oglop	Oglop#8007	-	-
Ravified	Ravified#4133	-	Mod
Zupori	redwoods#0108	-	-
roastyb	roastyb#3775	@roastyb1	-
Sens0ryYard	Sens0ryYard#6988	-	-
shibacrypt	shibacrypt#2756	-	-
Snorlax	Snorlax#5376	-	-
SpittingNickels	SpittingNickels#7142	-	-
Tumbledyer	Tumbledyer#8408	@tumbdenomics	Dev
UnphaZeD	UnphaZeD#4758	-	-
wivern	wivern#0410	-	Dev
wwp	wwp#2782	-	-

## Contributors

Beefy is a community of contributors. Below is a list of Cowmoonity members with an active role.

Name	Discord	Primary Role
MrTitoune	MrTitoune#9043	Dev
Timelock-Watcher	Timelock-Watcher#9763	Dev
52x13	52x13#5502	Cowmoonity Hero
Agiziga	Agiziga#1264	Cowmoonity Hero
Appel   Gear	Appel#8795	Cowmoonity Hero / Gear
armads (Ark, Ark)	armads#1337	Cowmoonity Hero / Mod
BifiPeter	crypto_peter#0412	Cowmoonity Hero
jackgale.eth	jackgale.eth#3663	Cowmoonity Hero
MisterDollahSignz	MisterDollahSignz#6345	Cowmoonity Hero / Media
Sim One	Sim One#9602	Cowmoonity Hero / Mod
Terry (TheTopDefi.com)	terrys999#6597	Cowmoonity Hero
ZukoWick (AVAX, Maxi)	ZukoWick#2604	Cowmoonity Hero / Mod
cl 	CL#8898	Mod
dcFX0	dcFX0#9639	Mod
TA	TA Tokenhalter#2694	Mod
ab.0x	ab.0x#3716	Strategist
ABeely	ABeely#0760	Strategist
Cartman	cartman#6481	Strategist
cmdrkeen.eth	CmdrKeen#2408	Strategist
Frog	geschnarkus#8191	Strategist
Galmoli	Galmoli#9531	Strategist
Hym	Hym#6244	Strategist
jaxx	jaxx#5948	Strategist
Marth007	Marth007#4920	Strategist
Matty	Matty#0005	Strategist
plainview.eth	plainview.eth#0643	Strategist
pumpingGhost	pumpingGhost#6020	Strategist
Qkyrie	Qkyrie#7908	Strategist
shatterproof	shatterproof#0001	Strategist

## Unique contributors

As of 09-02-2023, there are 101 unique contributors among the following roles:

- 17 @Core
- 18 @Dev
- 23 @Strategist
- 47 @Beefy OGs
- 29 @Cowmoonity Heroes
- 27 @Cowmoonity Farmhand
- 13 @Mod

# Partnerships

Looking for a collaboration with Beefy? All the info you'll need is right here.

Beefy's network of partners and collaborators from across the world of DeFi are at the centre of what we do as a protocol. As a longstanding and trusted name in multiple DeFi markets, there are very few types of projects that we haven't collaborated with (though we're always keen to add more!). This page overviews our partnerships team and efforts, as well as the process that we have in place to handle requests from prospective partners.

## The Beefy Partnerships Team

Our relationships with existing and new partners are managed by our dedicated partnerships team. The key members of the team are:

- Weso - Head of Strategic Partnerships
- Frondoto - Partnerships Lead
- Zapmore - Partnerships Lead
- The Beefy Cow (TBC) - Social Media / Marketing Lead
- Cowmoonity Heroes and Farmhands - partnership vetting assistants

Even with our dedicated team, we're still inundated with partnerships requests. To manage this, we ask that prospective partners follow the process outlined below to kick things off, rather than reaching out directly to the team.

## How do I connect with Beefy?

Prospective partners should head to the #🐮-partnership-vetting channel on the [Beefy Discord server](#), to introduce yourself to our Cowmoonity. The channel is a dedicated space to discuss new and existing partnership opportunities, and for prospective partners to introduce themselves and their projects to our Cowmoonity.

Alternatively, if you prefer not to reach out via Discord, you can contact us through any of our other social media channels (including [Twitter](#), [Reddit](#) and [Telegram](#)), and we'll put you in touch with our vetting assistants.

## What does vetting consist of?

Our vetting process doesn't follow any strict guidelines or procedures, but simply involves our members gathering information from you and presenting this back to the Core team and Cowmoonity, together with a decision on whether to refer you to our Core team. Our members will look to assess the size and growth of your project or organisation, the nature and fit of your products and the kind of partnerships that could be pursued. We gladly accept projects volunteering their own information (check the channel for past examples), though our members will aim to come to their own verdict of whether to refer your request to Core.

To keep the process above board, we kindly ask that you refrain from directly messaging any of our Core team or Cowmoonity Heroes or Farmhands, unless specifically invited to. We're proud that our partnerships process embraces transparency, and hope that public vetting will provide you a platform to introduce your project to our Cowmoonity.

## Can't I just speak to Core?

We get it. Sometimes a request may feel too pressing or a project may feel too important to need to go through vetting. Or you may have tried our vetting process, but found that our assistants are standing in your way. You might think it's a good idea to start messaging our Core team directly, to make sure your request is heard... but please, don't.

Our Cowmoonity Heroes and Farmhands are longstanding friends of the DAO, and have our best interests at heart. They're also well aware of when a request *really* is important, and won't hesitate to put you straight in touch with Core if appropriate. However, taking matters into your own hands is likely to pull Beefy's key resources away from their hard work building the protocol, and isn't the best first impression if you're then referred back to vetting. So take a deep breath, smile and trust in the process... our vetting assistants are here to help you.

## What kinds of partnerships are there?

Beefy's network of partners spans every corner of the DeFi world, and beyond. There are really no limits to what a partnership with Beefy could entail. With that said, there are a number of common types of partnerships that we regularly engage in:

- Blockchain partners - add Beefy to your chain to build out its DeFi ecosystem.
- Decentralised exchange partners - bring liquidity to your exchange by having Beefy build autocompounding vaults on top of your farms.
- Protocol co-marketing partners - market your project with Beefy's vaults and launchpools.
- Wallet integration partners - have Beefy integrate your wallet into our site.
- Insurance partners - protect and market to Beefy's users with dedicated products.
- Infrastructure and security partners - bring your technology and services to bolster Beefy's operations.

## Any last tips for prospective partners?

Now that you mention it:

- Have a look through our #🐮-partnership-vetting channel on Discord, as well as all the threads attached to it. This'll give you a sense of how the vetting process goes, and what kind of information you'll be asked to give.
- If you want to give a really good first impression, perhaps consider preparing your own summary in the style adopted by our vetting assistants for previous requests. Our assistants will still want to double check what you say, but this may save both you and us a fair amount of time and effort.
- We'll often want to know about what you want Beefy to bring to a partnership, and what your project proposes to bring. Some points you may want to consider ahead of time are:
  - For common requests, like integrating a wallet into our site, we simply can't say yes to everyone who asks. So explain what separates you from others in the market, or what you plan to bring to Beefy to make an additional integration worth our while.
  - Where you're asking for co-marketing, bear in mind the relative size of your project and ours. For Beefy, partnering with tiny fledgling protocols often means we're giving more than we're getting in marketing efforts.
  - If you're considering asking Beefy's dev team to help you build your product, it's worth thinking about: (1) is your product a better use of their time than continuing to build our existing and successful protocol? and (2) if Beefy's team did want to build that product, why would they want or need to do it with you rather than doing it themselves?
- For projects with your own token, consider whether you may want a Beefy [vault](#) to attract liquidity to your project, and whether a [launchpool](#) partnership may be of benefit to you.
- If you're interested in a Beefy vault for your project, we'd recommend reading through our [SAFU practices](#) page first, and assess whether you meet our criteria. This is one of the first questions you'll be asked in vetting, so best to come prepared with a "yes"!
- If you're an individual looking to get involved with Beefy, you don't need a formal partnership! Head to the #🐮-we-are-hiring page on the Beefy Discord to see what kind of paid and volunteer roles are available. And get stuck in with us from there.

# Vault Contract

Last Update: February 2023

The [Beefy Vault Contract](#) is the central user-facing implementation of the Beefy protocol, which accepts and manages user deposits and mints mooTokens as a proof of receipt to facilitate withdrawals. It follows the ERC-20 [standard](#) for fungible, transferrable tokens.

Besides handling deposits and withdrawals, the primary function of the vault is to direct deposited funds to the relevant autocompounding [Strategy Contract](#). The vault and strategy contracts are kept separate to isolate any risks in the strategy from user deposits.

## View Functions

### want()

Returns the address of the underlying farm token (e.g. the LP token) used in both the Beefy Vault and Strategy contracts. Note that this is not the same as the underlying assets used for the farm.

```
function want() public view returns (IERC20Upgradeable) {
    return IERC20Upgradeable(strategy.want());
}
```

### balance()

Returns the amount of "want" (i.e. underlying farm token) stored in the vault and strategy and yield source as an integer.

```
function balance() public view returns (uint) {
    return want().balanceOf(address(this)) + IStrategyV7(strategy).balanceOf();
}
```

### available()

Returns the amount of "want" (i.e. underlying farm token) stored in the vault alone as an integer.

```
function available() public view returns (uint256) {
    return want().balanceOf(address(this));
}
```

### totalSupply()

Returns the total amount of mooTokens minted as an integer, which are always displayed as 18 decimal token. This is a standard method inherited from the ERC-20 standard. See [# What are mooTokens?](#) for more details.

```
function totalSupply() public view virtual override returns (uint256) {
    return _totalSupply;
}
```

### getPricePerFullShare()

Returns the current price per share of the vault (i.e. per mooToken) as an integer denominated in the "want" (i.e. underlying farm token). This is calculated as *Price per Full Share* = **# balance()** / **# totalSupply()**.

```
function getPricePerFullShare() public view returns (uint256) {
    return totalSupply() == 0 ? 1e18 : balance() * 1e18 / totalSupply();
}
```

### strategy()

Returns the address current underlying strategy contract that the vault is using to generate yield.

```
function strategy() external view returns (address);
```

## Write Functions

### deposit()

Executes a transfer of a specified `_amount` of "want" (i.e. underlying farm token) from the depositor to the vault, and then mints a proportional quantity of mooTokens to the depositor in return.

```
function deposit(uint _amount) public nonReentrant {
    strategy.beforeDeposit();
    uint256 _pool = balance();
    want().safeTransferFrom(msg.sender, address(this), _amount);
    earn();
    uint256 _after = balance();
    _amount = _after - _pool; // Additional check for deflationary tokens
    uint256 shares = 0;
    if (totalSupply() == 0) {
        shares = _amount;
    } else {
        shares = (_amount * totalSupply()) / _pool;
    }
    _mint(msg.sender, shares);
}
```

Additionally, there is a helper function *depositAll()* that deposits the entire balance of "want" in the user's wallet at the time of the transaction.

### withdraw()

Executes a burn of a specified `_amount` of mooTokens from the depositor, and then transfers a proportional quantity of "want" (i.e. underlying farm token) to the depositor in return.

```
function withdraw(uint256 _shares) public {
    uint256 r = (balance() * _shares) / totalSupply();
    _burn(msg.sender, _shares);
    uint b = want().balanceOf(address(this));
    if (b < r) {
        uint _withdraw = r - b;
        strategy.withdraw(_withdraw);
        uint _after = want().balanceOf(address(this));
        uint _diff = _after - b;
        if (_diff < _withdraw) {
            r = b + _diff;
        }
    }
    want().safeTransfer(msg.sender, r);
}
```

Similarly to [# deposit\(\)](#), there is a helper function *withdrawAll()* that withdraw the entire balance of mooTokens in the user's wallet at the time of the transaction.

### earn()

Executes a transfer of [# available\(\)](#) "want" (i.e. underlying farm token) from the Vault Contract to the strategy contract and triggers the strategy's *deposit()* function to deploy the funds and begin earning.

```
function earn() public {
    uint _bal = available();
    want().safeTransfer(address(strategy), _bal);
    strategy.deposit();
}
```

### proposeStrat()

Writes the address of an alternate strategy to the Vault Contract's memory, in anticipation of upgrade the current strategy to the alternate using [# upgradeStrat\(\)](#).

```
function proposeStrat(address _implementation) public onlyOwner {
    require(address(this) == IStrategyV7(_implementation).vault(), "Proposal not valid");
    require(want() == IStrategyV7(_implementation).want(), "Different want");
    stratCandidate = StratCandidate({
        implementation: _implementation,
        proposedTime: block.timestamp
    });
    emit NewStratCandidate(_implementation);
}
```

### upgradeStrat()

Replaces the address of the current strategy with an alternate strategy specified by [# proposeStrat\(\)](#).

```
function upgradeStrat() public onlyOwner {
    require(stratCandidate.implementation != address(0), "There is no candidate");
    require(stratCandidate.proposedTime + approvalDelay < block.timestamp, "Delay has not passed");
    emit UpgradeStrat(stratCandidate.implementation);
    strategy.retireStrat();
    strategy = IStrategyV7(stratCandidate.implementation);
    stratCandidate.implementation = address(0);
    stratCandidate.proposedTime = 5000000000;
    earn();
}
```

## BeefyVaultV7.sol

The current release of our standard Beefy Vault Contract is [BeefyVaultV7.sol](#), which was released in August 2022. The V7 release improved on the previous version in a few keys ways:

- Introduced vault upgradeability through proxy patterns, to facilitate updates and changes to live Beefy vaults without needing to deprecate and re-deploy;
- Updated the strategy interface to allow for upgradeable strategies; and
- Amended all contracts to remove reliance on the SafeMath library, which has been generally retired following incorporation of its features into Solidity v0.8.

Separately, a ERC-4646-compliant wrapper contract was released for the V7 vault in November 2022, which allows developers to incorporate Beefy Vaults into their projects with standardised vault functionality and interfaces. See [BeefyWrapper Contract](#) for more information.

# Strategy Contract

Last Update: February 2023

Strategy Contracts are the primary driver of Beefy's investment model, which facilitate the auto-compounding of yield farm rewards. Beefy's process has three key steps: (1) staking deposited tokens in the relevant farms, (2) harvesting rewards, and (3) swapping rewards for more deposit tokens and reinvesting the proceeds.

Each strategy contract is ultimately dependent on a [Vault Contract](#) for the capital they deploy, and do not have any direct interaction with ordinary users. The vault and strategy contracts are kept separate to isolate any risks in the strategy from user deposits.

## Dependencies

All Beefy strategies rely on a range of dependencies and interfaces which are imported into the strategy contract on deployment. The core dependencies, which allow the strategy to inherit a range of functionality are:

- the [StratFeeManager Contract](#); and
- the [GasFeeThrottler Contract](#).

## Interfaces

The key interfaces which allow the strategy to interact with third party contracts are:

- the **router contract interface** - which allows for swaps between the different tokens involved in the auto-compounding process (e.g. `ILiwiSwapRouterETH.sol`);
- the **liquidity pool contract interface** - which is the underlying pool that our vaults provide liquidity to and that the farms are built on top of (e.g. `ILiwiSwapV2Pair.sol`); and
- the **chef contract interface** - the farm which is issuing rewards for providing liquidity (e.g. `IMiniChefV2.sol`).

## View Functions

### balanceOf() / balanceOfWant()

Checks the amount of the underlying farm token (or "want") stored in the strategy. Returns the specific amount of tokens.

```
function balanceOf() public view returns (uint256) {
    return balanceOfWant() + balanceOfPool();
}

function balanceOfWant() public view returns (uint256) {
    return IERC20(want).balanceOf(address(this));
}
```

### balanceOfPool()

Checks the amount of underlying farm token (or "want") stored in the chef contract. Returns the specific amount of tokens.

```
function balanceOfPool() public view returns (uint256) {
    (uint256 _amount, ) = IMiniChefV2(chef).userInfo(poolId, address(this));
    return _amount;
}
```

### rewardsAvailable()

Checks the amount of pending rewards held by the chef contract capable of being claimed by the strategy contract. Returns the specific amount of tokens.

```
function rewardsAvailable() public view returns (uint256) {
    return IMiniChefV2(chef).pendingSushi(poolId, address(this));
}
```

### callReward()

Most strategies include a `callReward()` function, which is used to determine the amount of "native" token rewards available to the `harvest()` caller.

```
function callReward() external view returns (uint256) {
    uint256 pendingReward;
    address rewarder = IMiniChefV2(chef).rewarder(poolId);
    if (rewarder != address(0)) {
        pendingReward = IRewarder(rewarder).pendingToken(poolId, address(this));
    }
    uint256 outputBal = rewardsAvailable();
    uint256 nativeOut;
    if (reward == native) {
        nativeOut = pendingReward;
    } else if (pendingReward > 0) {
        uint256 poolLength = params.rewardToNative_path.length;
        uint256 amount = pendingReward;
        for (uint i; i < poolLength; i++) {
            bytes memory data = abi.encode(routes.rewardToNative[i], amount);
            amount = IIdentPool(params.rewardToNative_path[i].pool).getAmountOut(data);
            unchecked { ++i; }
        }
        nativeOut = amount;
    }
    if (outputBal > 0) {
        bytes memory data = abi.encode(output, outputBal);
        nativeOut += IIdentPool(params.outputToNative_path[0].pool).getAmountOut(data);
    }
    IFeeConfig.FeeCategory memory fees = getFees();
    return nativeOut * fees.total / DIVISOR * fees.call / DIVISOR;
}
```

## Write Functions

### deposit()

Deposits the underlying farm token (or "want") into the farm by way of the connected chef contract. First checks that the strategy is holding some of the underlying farm token (or "want") before depositing the entire balance in the chef.

```
function deposit() public whenNotPaused {
    uint256 wantBal = IERC20(want).balanceOf(address(this));
    if (wantBal > 0) {
        IMiniChefV2(chef).deposit(poolId, wantBal, address(this));
        emit Deposit(balanceOf());
    }
}
```

### withdraw()

External function called by the vault to facilitate user withdrawals. First checks that the balance of the underlying farm token (or "want") is sufficient to fulfil the request, and then withdraws that amount from the chef contract, before transferring back to the vault contract.

```
function withdraw(uint256 _amount) external {
    require(msg.sender == vault, "Ivault");
    uint256 wantBal = IERC20(want).balanceOf(address(this));
    if (wantBal < _amount) {
        IMiniChefV2(chef).withdraw(poolId, _amount.sub(wantBal), address(this));
        wantBal = IERC20(want).balanceOf(address(this));
    }
    if (wantBal > _amount) {
        wantBal = _amount;
    }
    if (tx.origin != owner() && !paused()) {
        uint256 withdrawalFeeAmount = wantBal * withdrawalFee / WITHDRAWAL_MAX;
        wantBal = wantBal - withdrawalFeeAmount;
    }
    IERC20(want).safeTransfer(vault, wantBal);
    emit Withdraw(balanceOf());
}
```

### harvest()

Harvest invokes the compounding of the vault for all users. Specifically, this harvests from the chef contract, charges fees on the harvest and then deposits the harvested rewards back into the farm to achieve the auto-compounding effect.

This function is completely decentralized, meaning that anyone is able to call the function, and can earn a reward between 0.05 - 0.5% of the total yield. This can be called by any one of three methods, detailed below.

```
// @dev Default harvest() method.
function harvest() external virtual gasThrottle {
    _harvest(tx.origin);
}

// @dev Alternative harvest() method, where caller receives a fee.
function harvest(address callFeeRecipient) external virtual {
    _harvest(callFeeRecipient);
}

// @dev Alternative harvest() method, where manager calls without gas throttling.
function managerHarvest() external onlyManager {
    _harvest(tx.origin);
}

// @dev Underlying internal _harvest() function, used by all 3 public methods.
function _harvest(address callFeeRecipient) internal whenNotPaused {
    IMiniChefV2(chef).harvest(poolId, address(this));
    uint256 outputBal = IERC20(output).balanceOf(address(this));
    uint256 rewardBal = IERC20(reward).balanceOf(address(this));
    if (outputBal > 0 || rewardBal > 0) {
        chargeFees(callFeeRecipient);
        addLiquidity();
        uint256 wantHarvested = balanceOfWant();
        deposit();
        lastHarvest = block.timestamp;
        emit StratHarvest(msg.sender, wantHarvested, balanceOf());
    }
}
```

### chargeFees()

Internal method to charge fees on every `harvest()` call, by swapping the native token in the strategy to the output token via the router contract. The contract then calculates the output for the different fee recipient and transfers the output tokens according to the allocation. The recipients are the harvest caller, the strategist who deployed the contract and the Beefy treasury.

```
function chargeFees(address callFeeRecipient) internal {
    IFeeConfig.FeeCategory memory fees = getFees();
    uint256 rewardBal = IERC20(reward).balanceOf(address(this));
    if (rewardBal > 0 && reward != native) {
        IRouter.ExactInputParams memory _rewardToNative =
            params.rewardToNative;
        _rewardToNative.amountIn = rewardBal;
        IRouter.ExactInputWithNativeToken(_rewardToNative);
    }
    uint256 outputBal = IERC20(output).balanceOf(address(this));
    if (outputBal > 0) {
        IRouter.ExactInputParams memory _outputToNative =
            params.outputToNative;
        _outputToNative.amountIn = outputBal;
        IRouter.ExactInputWithNativeToken(_outputToNative);
    }
    uint256 nativeBal = IERC20(native).balanceOf(address(this)) * fees.total /
        DIVISOR;
    uint256 callFeeAmount = nativeBal * fees.call / DIVISOR;
    IERC20(native).safeTransfer(callFeeRecipient, callFeeAmount);
    uint256 beefyFeeAmount = nativeBal * fees.beefy / DIVISOR;
    IERC20(native).safeTransfer(beefyFeeRecipient, beefyFeeAmount);
    uint256 strategistFeeAmount = nativeBal * fees.strategist / DIVISOR;
    IERC20(native).safeTransfer(strategist, strategistFeeAmount);
    emit ChargedFees(callFeeAmount, beefyFeeAmount, strategistFeeAmount);
}
```

### addLiquidity()

Internal method to add liquidity to the underlying pool for the farm as part of the `harvest()` function. Swaps the output token to the underlying tokens of the farm, and then adds both to the liquidity pool to obtain the underlying farm deposit token (or "want"). The remainder of the `harvest()` call then deposits these tokens in the farm.

```
function addLiquidity() internal {
    uint256 nativeBal = IERC20(native).balanceOf(address(this)) / 2;
    if (lpToken0 != native) {
        IRouter.ExactInputParams memory _nativeToLp0 = params.nativeToLp0;
        _nativeToLp0.amountIn = nativeBal;
        IRouter.ExactInputWithNativeToken(_nativeToLp0);
    }
    if (lpToken1 != native) {
        IRouter.ExactInputParams memory _nativeToLp1 = params.nativeToLp1;
        _nativeToLp1.amountIn = nativeBal;
        IRouter.ExactInputWithNativeToken(_nativeToLp1);
    }
    IRouter.TokenInput[] memory tokens = new IRouter.TokenInput[](2);
    uint256 lp0Bal = IERC20(lpToken0).balanceOf(address(this));
    uint256 lp1Bal = IERC20(lpToken1).balanceOf(address(this));
    tokens[0] = IRouter.TokenInput(lpToken0, true, lp0Bal);
    tokens[1] = IRouter.TokenInput(lpToken1, true, lp1Bal);
    bytes memory data = abi.encode(address(this));
    IRouter.ExactInputWithNativeToken(tokens, want, 1, data);
}
```

### setHarvestOnDeposit()

Most Beefy vaults `harvest` on deposit. This means that, before the user's funds enter the strategy, the yield on the entire vault is harvested and reinvested. This prevents new depositors from stealing the yield of existing depositors. As a result, any vault that is set to harvest on deposit is able to remove the withdrawal fee completely.

`harvestOnDeposit` is a boolean variable which is set to true when the vault is harvesting on deposit. This is toggled by the `setHarvestOnDeposit()` function, set out below:

```
bool public harvestOnDeposit;

function setHarvestOnDeposit(bool _harvestOnDeposit) external onlyManager {
    harvestOnDeposit = _harvestOnDeposit;
    if (harvestOnDeposit) {
        setWithdrawalFee(0);
    } else {
        setWithdrawalFee(10);
    }
}
```

### beforeDeposit()

External function used to facilitate harvests on deposit, if active. Checks first that harvest on deposit is active and that the caller is the vault, before harvesting.

```
function beforeDeposit() external override {
    if (harvestOnDeposit) {
        require(msg.sender == vault, "Ivault");
        _harvest(tx.origin);
    }
}
```

### panic()

Beefy does not directly touch any user funds held in the protocol. During times of uncertainty or upgrades to the underlying yield farm, Beefy can withdraw all funds out of third party contracts and hold them safely in the strategy using the `panic()` function. By "panicking" the strategy, users remain able to withdraw their funds from the vault without any delay or exposure to third party risks. This function also removes all allowances to both the UniRouter and the underlying yield farm contract, to ensure no funds can be withdrawn by those contracts.

```
function panic() public onlyManager {
    pause();
    IMiniChefV2(chef).emergencyWithdraw(poolId, address(this));
}
```

### pause() / unpause()

All Beefy strategies are pausable, meaning that functionality can be halted during the strategy's ordinary operations by the strategy manager. This is inherited through `StratManager.sol`, and relies on the standard `Pausable` or `AbstractContract`. This function also removes all allowances to both the UniRouter and the underlying yield farm contract, to ensure no funds can be withdrawn by those contracts.

```
function pause() public onlyManager {
    _pause();
    _removeAllowances();
}
```

In the reverse, strategies can also be unpaused, by reversing the actions in the pause function.

```
function unpause() external onlyManager {
    _unpause();
    _giveAllowances();
    deposit();
}
```

The functions affected by a pause in most strategy contracts are `deposit()`, `withdraw()` and `harvest()`.

### \_giveAllowances() / \_removeAllowances()

Internal functions used to set and remove all allowances with third party contracts, to control whether third party contracts have the necessary permissions to withdraw funds from the strategy. The relevant contracts are the underlying farm token/`want()` (e.g. LP token), the strategy output token (often the same as the `want()`), the native chain token (used for gas) and the underlying tokens used for the farm.

```
function _giveAllowances() internal {
    IERC20(want).safeApprove(chef, type(uint).max);
    IERC20(output).safeApprove(unirouter, type(uint).max);
    IERC20(native).safeApprove(unirouter, type(uint).max);
    IERC20(lpToken0).safeApprove(unirouter, 0);
    IERC20(lpToken0).safeApprove(unirouter, type(uint).max);
    IERC20(lpToken1).safeApprove(unirouter, 0);
    IERC20(lpToken1).safeApprove(unirouter, type(uint).max);
}

function _removeAllowances() internal {
    IERC20(want).safeApprove(chef, 0);
    IERC20(output).safeApprove(unirouter, 0);
    IERC20(native).safeApprove(unirouter, 0);
    IERC20(lpToken0).safeApprove(unirouter, 0);
    IERC20(lpToken1).safeApprove(unirouter, 0);
}
```

### retireStrat()

External function used as part of a migration from one strategy to another. This effectively closes down the strategy by withdrawing all funds and transferring them back to the vault. It can only be triggered by a call from the vault contract.

```
function retireStrat() external {
    require(msg.sender == vault, "Ivault");
    IMiniChefV2(chef).emergencyWithdraw(poolId, address(this));
    uint256 wantBal = IERC20(want).balanceOf(address(this));
    IERC20(want).transfer(vault, wantBal);
}
```

# StratFeeManager Contract

Last Update: February 2023

The `StratFeeManager` Contract is collection of important dependencies which are imported and used in every Beefy `Strategy Contract`.

Originally, these dependencies were split into two contracts - `StratManager.sol` and `FeeManager.sol`. After the move to Solidity V0.8, the two were combined into a single contract - `StratFeeManager.sol`. The current version - `StratFeeManagerInitializable.sol` - facilitated a move to proxy clone strategies (which must be initialized with the relevant arguments for the strategy and its dependencies), to avoid the need to deploy every single strategy individually.

## Dependencies

The `StratFeeManager` contracts also introduce additional dependencies themselves, specifically `Ownable.sol` - which introduces functionality to set a contract's owner and restrict functionality to them alone - and `Pausable.sol` - which introduces functionality to freeze functionality in a contract by putting the contract on pause. Both dependencies are ultimately included in every Beefy Strategy Contract.

## Modifiers

Introduces an `onlyManager()` modifier to constrain functions to use by the strategy manager only.

```
modifier onlyManager() {
    require(msg.sender == owner() || msg.sender == keeper, "!manager");
    _;
}
```

## View Functions

### getFees()

Returns the value of all fees from the fee configuration contract.

```
function getFees() internal view returns (IFeeConfig.FeeCategory memory) {
    return beefyFeeConfig.getFees(address(this));
}
```

### getAllFees()

Returns the value of all fees from the fee configuration contract, as well as the dynamic deposit and withdrawal fees.

```
function getAllFees() external view returns (IFeeConfig.AllFees memory) {
    return IFeeConfig.AllFees(getFees(), depositFee(), withdrawFee());
}
```

### getStratFeeId()

Returns the integer value of the strategy fee ID from the fee configuration contract.

```
function getStratFeeId() external view returns (uint256) {
    return beefyFeeConfig.stratFeeId(address(this));
}
```

## Write Functions

### setStratFeeId()

Sets a new integer value for the strategy's fee ID, which indicates the relevant fee for the strategy.

```
function setStratFeeId(uint256 _feeId) external onlyManager {
    beefyFeeConfig.setStratFeeId(_feeId);
    emit SetStratFeeId(_feeId);
}
```

### setWithdrawalFee()

Sets a new integer value for the contract's withdrawal fee which is charged on each harvest.

```
function setWithdrawalFee(uint256 _fee) public onlyManager {
    require(_fee <= WITHDRAWAL_FEE_CAP, "!cap");
    withdrawalFee = _fee;
    emit SetWithdrawalFee(_fee);
}
```

### setVault()

Sets a new address for the contract's vault, which manages user funds.

```
function setVault(address _vault) external onlyOwner {
    vault = _vault;
    emit SetVault(_vault);
}
```

### setUnirouter()

Sets a new address for the contract's router which processes swaps within the contract.

```
function setUnirouter(address _unirouter) external onlyOwner {
    unirouter = _unirouter;
    emit SetUnirouter(_unirouter);
}
```

### setKeeper()

Sets a new address for the contract's keeper, who can "panic" the strategy.

```
function setKeeper(address _keeper) external onlyManager {
    keeper = _keeper;
    emit SetKeeper(_keeper);
}
```

### setStrategist()

Sets a new address for the contract's strategist who receives the strategist fee.

```
function setStrategist(address _strategist) external {
    require(msg.sender == strategist, "!strategist");
    strategist = _strategist;
    emit SetStrategist(_strategist);
}
```

### setBeefyFeeRecipient()

Sets a new address for the recipient of Beefy's fee on harvests, typically a Beefy treasury contract.

```
function setBeefyFeeRecipient(address _beefyFeeRecipient) external onlyOwner {
    beefyFeeRecipient = _beefyFeeRecipient;
    emit SetBeefyFeeRecipient(_beefyFeeRecipient);
}
```

### setBeefyFeeConfig()

Sets a new address for the fee configuration contract used by the strategy to fetch fees.

```
function setBeefyFeeConfig(address _beefyFeeConfig) external onlyOwner {
    beefyFeeConfig = IFeeConfig(_beefyFeeConfig);
    emit SetBeefyFeeConfig(_beefyFeeConfig);
}
```

# GasFeeThrottler Contract



Last Update: February 2023

The [GasFeeThrottler Contract](#) (formerly GasThrottler) is a smart contract device used to ensure that gas prices for child contract transactions always fall below a fixed maximum, or otherwise causes the transaction to be reverted. To do so, it points to a specific [GasPrice Contract](#), where the maximum gas price can be configured by the contract's owner. The GasFeeThrottler is incorporated into every [Strategy Contract](#).

## GasFeeThrottler.sol

The throttler contract has three main elements which are inherited by its child contracts:

1. the **shouldGasThrottle variable** - which is boolean fixed to "true" to indicate that a child contract has inherited gas throttling;
2. the **gasPrice variable** - which connects the throttler to `# GasPrice.sol` to identify the max gas price fixed by that contract; and
3. the **gasThrottle() modifier** - which requires that gas prices for transactions arising from the child contract's modified functions always are equal to or below the fixed max gas price.

```
contract GasFeeThrottler {  
  
    bool public shouldGasThrottle = true;  
  
    address public gasprice = address(0xA43509661141F254F54D9A326E8Ec851A0b95307);  
  
    modifier gasThrottle() {  
        if (shouldGasThrottle && Address.isContract(gasprice)) {  
            require(tx.gasprice <= IGasPrice(gasprice).maxGasPrice(), "gas is too high!");  
        }  
        -;  
    }  
}
```

## GasPrice.sol

The GasPrice contract provides three core elements to facilitate its purpose:

1. the **maxGasPrice variable** - which stores the current maximum gas price value set by the contract, and can be called externally using the [IGasPrice.sol](#) interface;
2. a **NewMaxGasPrice event** - which is triggered on changes to the maxGasPrice variable, and returns the old and new prices for external reference; and
3. the **setMaxGasPrice function** - which accepts an integer value for the new \_maxGasPrice as an argument, emits the NewMaxGasPrice event and updates the maxGasPrice variable.

```
contract GasPrice is Ownable {  
  
    uint public maxGasPrice = 10000000000;  
  
    event NewMaxGasPrice(uint oldPrice, uint newPrice);  
  
    function setMaxGasPrice(uint _maxGasPrice) external onlyOwner {  
        emit NewMaxGasPrice(maxGasPrice, _maxGasPrice);  
        maxGasPrice = _maxGasPrice;  
    }  
}
```

# FeeConfigurator Contract

Last Update: February 2023

The `BeefyFeeConfigurator Contract` is an infrastructure contract hosted on each of the blockchains which Beefy has deployed on. The contract manages the configuration of fees for each strategy on the relevant chain, which the `StratFeeManager Contract` interfaces with through `IFeeConfig.sol`.

The relevant address for the FeeConfigurator Contract ("*beefyFeeConfig*") on each chain is displayed in the Beefy API using the `#GET /config` endpoint.

## Modifier

Includes a standard `onlyManager()` modifier to control access to write functions.

```
modifier onlyManager() {
    require(msg.sender == owner() || msg.sender == keeper, "!manager");
    _;
}
```

## View Functions

### getFees()

Returns a `FeeCategory` structure for a specific strategy address argument, displaying the total fees charged, the fees for Beefy, the harvest caller and the strategist, a string showing the description of the type of fee category, and "active" boolean variable, showing whether the fee category is switched on or not.

Includes an "`_adjust`" boolean variable argument, which displays fees as a % of the total harvest if set to true, or as a % of the total fee if set to false.

```
function getFees(address _strategy) external view returns (FeeCategory memory) {
    return getFeeCategory(stratFeeId[_strategy], false);
}

function getFees(address _strategy, bool _adjust) external view returns (FeeCategory memory) {
    return getFeeCategory(stratFeeId[_strategy], _adjust);
}
```

### getFeeCategory()

Returns a `FeeCategory` structure for a specific strategy ID integer, as described above. Also includes the "`_adjust`" boolean variable option.

```
function getFeeCategory(uint256 _id, bool _adjust) public view returns (FeeCategory memory fees) {
    uint256 id = feeCategory[_id].active ? _id : 0;
    fees = feeCategory[id];
    if (_adjust) {
        uint256 _totalFee = fees.total;
        fees.beefy = fees.beefy * _totalFee / DIVISOR;
        fees.call = fees.call * _totalFee / DIVISOR;
        fees.strategist = fees.strategist * _totalFee / DIVISOR;
    }
}
```

## Write Functions

### setStratFeeId()

Updates the `stratFeeId` mapping to show ultimately what `FeeCategory` structure is being used by a particular strategy, by way of an intermediate `feeCategory` mapping and `feeId` integer values.

This includes 3 options, including one for the strategy to update its own `feeId`, one to stipulate the strategy address and `feeId` as arguments, and one to set a range of strategies by giving both an array of strategy addresses and an array of `feeIds` as arguments. Each of the three then use the internal `_setStratFeeId()` function to update each strategy.

```
function setStratFeeId(uint256 _feeId) external {
    _setStratFeeId(msg.sender, _feeId);
}

function setStratFeeId(address _strategy, uint256 _feeId) external onlyManager {
    _setStratFeeId(_strategy, _feeId);
}

function setStratFeeId(address[] memory _strategies, uint256[] memory _feeIds) external {
    uint256 stratLength = _strategies.length;
    for (uint256 i = 0; i < stratLength; i++) {
        _setStratFeeId(_strategies[i], _feeIds[i]);
    }
}

function _setStratFeeId(address _strategy, uint256 _feeId) internal {
    stratFeeId[_strategy] = _feeId;
    emit SetStratFeeId(_strategy, _feeId);
}
```

### setFeeCategory()

Sets the parameters for a given `FeeCategory` structure (new or existing), including the split in fees between Beefy, the harvest caller and the strategist.

```
function setFeeCategory(
    uint256 _id,
    uint256 _total,
    uint256 _call,
    uint256 _strategist,
    string memory _label,
    bool _active,
    bool _adjust
) external onlyOwner {
    require(_total <= totalLimit, ">totalLimit!");
    if (_adjust) {
        _call = _call * DIVISOR / _total;
        _strategist = _strategist * DIVISOR / _total;
    }
    uint256 beefy = DIVISOR - _call - _strategist;
    FeeCategory memory category = FeeCategory(_total, beefy, _call, _strategist, _label);
    feeCategory[_id] = category;
    emit SetFeeCategory(_id, _total, beefy, _call, _strategist, _label, _active);
}
```

### setKeeper()

Updates the named keeper in the FeeConfigurator contract.

```
function setKeeper(address _keeper) external onlyManager {
    keeper = _keeper;
    emit SetKeeper(_keeper);
}
```

### pause() / unpause()

Sets a specific `FeeCategory` (using the category ID as an argument) to either active ("unpaused") - meaning a strategy set to that category will use that category - or inactive ("paused") - meaning the strategy will revert to the default fee configuration.

```
function pause(uint256 _id) external onlyManager {
    feeCategory[_id].active = false;
    emit Pause(_id);
}

function unpause(uint256 _id) external onlyManager {
    feeCategory[_id].active = true;
    emit Unpause(_id);
}
```



# BeefyWrapper Contract

Last Update: February 2023

The BeefyWrapper contract is an ERC-4626 adapter interface that makes a Beefy [Vault Contract](#) compatible with the ERC-4626 standard. It unlocks the composability of the standard and enables smoother interfacing and interaction with Beefy Vaults by external protocols, without the need for additional adapters and plug-ins.

This page sets out some of the background to the ERC-4626 standard, and the functionality of the BeefyWrapper contract.

## Why ERC-4626?

The purpose of the ERC-4626 standard is to solve the problems caused by the diversity of vault designs found across DeFi. Many protocols have incorporated vault concepts into their architecture by reinventing the wheel to suit their own unique architecture and use cases. This means external projects hoping to implement a range of different vaults need to adapt and plug their code to reconcile it with the quirks of each protocol's unique vault design.

The new standard recognises some common functions across the majority of vaults, and suggests that harmonising this functionality into a single standard can help to mitigate the quantity of adaptations and plug-ins required to work with most vaults. For protocols like Beefy, adopting ERC-4626 for our Beefy Vaults helps to facilitate new product development building on our products, and as a result promises to increase the range of use cases available to our users.

## Contract Functions

The functionality of the BeefyWrapper contract facilitates the minting and burning of wrapped Beefy Vault tokens in exchange for the transfer of the caller's Beefy Vault tokens. It also overrides the standard deposit and withdraw functions to facilitate deposits to the main Beefy Vault, in exchange for minting and burning of the wrapped Beefy Vault tokens.

### wrap()

Transfers the specified amount of the caller's Beefy Vault tokens to the wrapper contract to mint the same quantity of wrapped Beefy Vault tokens to the caller.

```
// Wraps an amount of vault share tokens.
/// "amount" parameter is the total amount of vault share tokens to be wrapped.
function wrap(uint256 amount) public {

    // Transfers the specified amount of the caller's Beefy Vault tokens to the
    // wrapper.
    IERC20Upgradeable(vault).safeTransferFrom(msg.sender, address(this), amount);

    // Mints the specified amount wrapped Beefy Vault tokens to the caller.
    _mint(msg.sender, amount);
}
```

### wrapAll()

Utilises the wrap() function, but using the full balance of the caller as the "amount" parameter.

```
// Wraps all vault share tokens owned by the caller.
function wrapAll() external {
    wrap(IERC20Upgradeable(vault).balanceOf(msg.sender));
}
```

### unwrap()

Burns the specified amount of the caller's wrapped Beefy Vault tokens, to transfer the same quantity of unwrapped tokens from the wrapper contract back to the caller.

```
// Unwraps an amount of vault share tokens.
/// "amount" parameter is the total amount of vault share tokens to be unwrapped.
function unwrap(uint256 amount) public {

    // Burns the specified amount of the caller's wrapped Beefy Vault tokens.
    _burn(msg.sender, amount);

    // Transfers the specified amount of Beefy Vault tokens back to the caller.
    IERC20Upgradeable(vault).safeTransfer(msg.sender, amount);
}
```

### unwrapAll()

Utilises the unwrap() function, but using the full balance of the caller as the "amount" parameter.

```
// Unwraps all wrapped tokens owned by the caller.
function unwrapAll() external {
    unwrap(balanceOf(msg.sender));
}
```

### \_deposit()

Overrides the standard \_deposit() function to interact with the wrapper contract and issued wrapped Beefy Vault tokens, in place of the unwrapped version. Otherwise facilitates an ordinary transfer into the Beefy Vault.

```
// Deposit assets to the vault and mint an equal number of wrapped tokens to vault
// shares.
/// "caller" parameter is the address of the sender of the assets.
/// "receiver" parameter is the address of the receiver of the wrapped tokens.
/// "assets" parameter is the amount of assets being deposited.
/// "shares" parameter is the amount of shares being minted.
function _deposit(address caller, address receiver, uint256 assets, uint256 shares)
    internal virtual override {

    // Transfers the caller's tokens to the wrapper.
    IERC20Upgradeable(asset()).safeTransferFrom(caller, address(this), assets);
    uint balance = IERC20Upgradeable(vault).balanceOf(address(this));

    // Deposits the caller's tokens into the Beefy Vault.
    IVault(vault).deposit(assets);

    // Mints wrapped tokens to the receiver.
    shares = IERC20Upgradeable(vault).balanceOf(address(this)) - balance;
    _mint(receiver, shares);

    // Emits the Deposit event to signify a successful deposit.
    emit Deposit(caller, receiver, assets, shares);
}
```

### \_withdraw()

Overrides the standard \_withdraw() function to interact with the wrapped Beefy Vault tokens, in place of the unwrapped version. Otherwise facilitates an ordinary withdrawal from the Beefy Vault and returns the underlying tokens to the receiver.

```
// Burn wrapped tokens and withdraw assets from the vault.
/// "caller" parameter is the address of the caller of the withdraw.
/// "receiver" parameter is the address of the receiver of the assets.
/// "owner" parameter is the address of the owner of the burnt shares.
/// "assets" parameter is the amount of assets being withdrawn.
/// "shares" parameter is the amount of shares being burnt.
function _withdraw(address caller, address receiver, address owner, uint256 assets,
    uint256 shares) internal virtual override {

    // Checks caller is not the contract's owner, and that the caller's spend
    // allowance is sufficient for the call.
    if (caller != owner) {
        _spendAllowance(owner, caller, shares);
    }

    // Burns the caller's wrapped tokens.
    _burn(owner, shares);

    // Withdraws the caller's assets from the Beefy Vault.
    IVault(vault).withdraw(shares);
    uint balance = IERC20Upgradeable(asset()).balanceOf(address(this));
    if (assets > balance) {
        assets = balance;
    }

    // Transfers the caller's assets back to the receiver.
    IERC20Upgradeable(asset()).safeTransfer(receiver, assets);

    // Emits the Withdraw event to signify a successful withdrawal.
    emit Withdraw(caller, receiver, owner, assets, shares);
}
```

### totalAssets()

Overrides the standard totalAssets() function to fetch the total assets held by the vault.

```
// Fetches and returns the total assets as a uint256 value.
function totalAssets() public view virtual override returns (uint256) {

    return IVault(vault).balance();
}
```

### totalSupply()

Overrides the standard totalSupply() function to fetch the total shares issues by the vault.

```
// Fetches and returns the total vault shares as a uint256 value.
function totalSupply() public view virtual override(IERC20Upgradeable,
    IERC20Upgradeable) returns (uint256) {

    return IERC20Upgradeable(vault).totalSupply();
}
```

## BeefyWrapperFactory Contract

The BeefyWrapperFactory contract is a factory contract which provides a minimal proxy pattern for use in creating new Beefy Vault wrappers. A factory contract allows users to generate and deploy their own proxy contracts that all point to the same implementation contract, where the logic resides.

Through the BeefyWrapperFactory, BeefyWrapper contracts can be deployed for any vaults. The factory contract has only one key function - clone().

### clone()

Uses the OpenZeppelin standard proxy template ClonesUpgradeable.sol to create a proxy contract which is a clone of the BeefyWrapper contract.

```
// Creates a new Beefy Vault wrapper as a proxy of the template instance.
/// "_vault" parameter is the cloned Beefy Vault.
/// "proxy" return is the new proxied Beefy Vault wrapper.
function clone(address _vault) external returns (address proxy) {

    // Proxy is set as a clone of the instance of the BeefyWrapper contract.
    proxy = implementation.clone();

    // Initializes the wrapper proxy set above.
    IWrapper(proxy).initialize(
        _vault,
        string.concat("W", IVault(_vault).name()),
        string.concat("w", IVault(_vault).symbol())
    );

    // Emits the ProxyCreated event to signify a successful deployment.
    emit ProxyCreated(proxy);
}
```

## Contracts

The template Beefy Vault wrapper contracts are publicly maintained in Beefy's GitHub repositories. See BeefyWrapper.sol and BeefyWrapperFactory.sol.

BeefyWrapperFactory.sol contracts for each Beefy Vault have been deployed separately on the relevant chains at the following addresses:

Chain	Text
Arbitrum	0x48bF3a071098a09C7D00379b4DBC69Ab6Da83a36
Aurora	Not yet deployed
Avalanche	0x1Fa046d28FF749b9D7CF7E9a41BEecd1260F11eD
BSC	0x85B792C67cEe281064eb7A3AF0Fe2A76E9a7849e
Canto	Not yet deployed
Celo	Not yet deployed
Cronos	Not yet deployed
Emerald	Not yet deployed
Ethereum	Not yet deployed
Fantom	0x985CABC1B4FF5a15E1162BaE1669A928e5a6bD49
Fuse	Not yet deployed
Harmony	Not yet deployed
Heco	Not yet deployed
Kava	Not yet deployed
Metis	0xDf29382141059afD25deb624E6c8f13A051012Be
Moonbeam	Not yet deployed
Moonriver	Not yet deployed
Optimism	0x182be93E1C0C4d305fe43bD093292F21f6679797
Polygon PoS	0x7e778f4cF8c7C43FB2F3C9c0b4c7CB7c2bad978
Polygon zkEVM	Not yet deployed
zkSync	Not yet deployed

# GaugeStaker Contract

Last Update: June 2022

## What is the GaugeStaker?

The [GaugeStaker contract](#) is the central smart contract for the [binSPIRIT](#) token, which manages both the collection of SpiritSwap protocol revenue for inSPIRIT holders and the delivery of inSPIRIT farm boosts to other Beefy SpiritSwap vaults.

## How does the GaugeStaker work?

The GaugeStaker has three notable roles: (1) managing user SPIRIT deposits to accumulate inSPIRIT rewards; (2) managing voting for SpiritSwap boosted farms; and (3) passing tokens between other Beefy SpiritSwap strategies and boosted farming gauges.

When users deposit SPIRIT, the GaugeStaker will automatically stake the SPIRIT with SpiritSwap to receive non-transferrable inSPIRIT. All staked SPIRIT must also be locked (so cannot be withdrawn), so the GaugeStaker locks all deposits for the longest possible period of time (currently 4 years) to receive the maximum amount of inSPIRIT. Protocol revenue rewards accrue constantly on the locked SPIRIT, and the GaugeStaker automatically claims these rewards and returns them to the [Beefy binSPIRIT vault](#) on a regular basis, where they are automatically compounded.

Holders of inSPIRIT are also entitled to [vote](#) in SpiritSwap governance and for the distribution of boosted farm rewards, so the GaugeStaker manages the allocation of these votes.

Finally, inSPIRIT holders also received boosted rewards on selected SpiritSwap farms. All boosted Beefy SpiritSwap vaults are configured to pass all of their deposits, withdrawals and harvesting of rewards through the GaugeStaker, to receive the benefits of the boost. This provides the highest possible boost across each of the farms, as the GaugeStaker holds the full concentration of Beefy's accumulated inSPIRIT.

## GaugeStaker Functionality

The GaugeStaker contract incorporates a range of different functionality and methods to execute its two roles. These include:

### Deposit SPIRIT to mint binSPIRIT

A user can deposit SPIRIT ( `want` ) and the contract will confirm the amount that is received by checking balances before and after the transfer. If the received amount is non-zero then check if an existing lock for SPIRIT exists, which it likely will unless the lock has not been initiated before or has been left to expire. If the lock exists then it will be extended out to the full 4 years if the current lock time is less than the full amount, and the received balance of SPIRIT is locked to get a 1:1 amount of inSPIRIT. If no lock currently exists then create a new one and lock the balance of SPIRIT on the contract. Finally mint an equal amount of binSPIRIT as the received balance of SPIRIT from the user.

```
// deposit 'want' and lock
function _deposit(address _user, uint256 _amount) internal nonReentrant whenNotPaused {
    uint256 _pool = balanceOfWant();
    want.safeTransferFrom(msg.sender, address(this), _amount);
    uint256 _after = balanceOfWant();
    _amount = _after.sub(_pool); // Additional check for deflationary tokens
    if (_amount > 0) {
        if (balanceOfVe() > 0) {
            increaseUnlockTime();
            veWant.increase_amount(_amount);
        } else {
            _createLock();
        }
        _mint(_user, _amount);
        emit DepositWant(balanceOfVe());
    }
}
```

### Vote on which gauges to boost

The Beefy Keeper can vote on gauge incentives using the inSPIRIT balance on the GaugeStaker as voting power. It will be mainly used to vote for Beefy and strategic partner's gauges, and can be governed by Beefy DAO to vote for various incentives on gauges. The voting function is a simple call to SpiritSwap's Gauge Proxy contract which records the votes and decides the distribution of gauge incentives. The Beefy keeper can split the voting power between multiple gauges in a single call using the parameter arrays.

```
// vote on boosted farms
function vote(address[] calldata _tokenVote, uint256[] calldata _weights) external onlyKeeper {
    gaugeProxy.vote(_tokenVote, _weights);
    emit Vote(_tokenVote, _weights);
}
```

### Pass through tokens between strategies and gauges

Strategies for Beefy's SpiritSwap vaults must pass through their deposits, withdrawals and harvests to and from the GaugeStaker. Only a whitelisted strategy can interact with the GaugeStaker, and each gauge is assigned at most one strategy.

Deposits and withdrawals pass through the exact amount that is requested ( `_amount` ) in the token that is assigned to the gauge ( `_underlying` ). Harvests ( `claimGaugeReward()` ) pass through only the SPIRIT ( `want` ) reward that's received by the GaugeStaker when claiming the reward, ignoring the existing balance on the GaugeStaker. None of the funds are kept on the GaugeStaker, they are always passed across in the same transaction.

```
// pass through a deposit to a gauge
function deposit(address _gauge, uint256 _amount) external onlyWhitelist(_gauge) {
    address _underlying = IGauge(_gauge).TOKEN();
    IERC20Upgradeable(_underlying).safeTransferFrom(msg.sender, address(this), _amount);
    IGauge(_gauge).deposit(_amount);
}

// pass through a withdrawal from a gauge
function withdraw(address _gauge, uint256 _amount) external onlyWhitelist(_gauge) {
    address _underlying = IGauge(_gauge).TOKEN();
    IGauge(_gauge).withdraw(_amount);
    IERC20Upgradeable(_underlying).safeTransfer(msg.sender, _amount);
}

// pass through rewards from a gauge
function claimGaugeReward(address _gauge) external onlyWhitelist(_gauge) {
    uint256 _before = balanceOfWant();
    IGauge(_gauge).getReward();
    uint256 _balance = balanceOfWant().sub(_before);
    want.safeTransfer(msg.sender, _balance);
}
```

### Claim SpiritSwap protocol fees

Holding inSPIRIT gives the GaugeStaker the right to claim a portion of SpiritSwap's protocol fees, which will be distributed to binSPIRIT stakers in a reward pool. The protocol fees are distributed once a week in the form of SPIRIT and need to be claimed from the Fee Distributor contract. The Reward Pool contract will call the claim function via `claimVeWantReward()`. Only the SPIRIT ( `want` ) reward is immediately passed back to the Reward Pool, if there is anything available to claim.

```
// pass through rewards from the fee distributor
function claimVeWantReward() external onlyRewardPool {
    uint256 _before = balanceOfWant();
    feeDistributor.claim();
    uint256 _balance = balanceOfWant().sub(_before);
    want.safeTransfer(msg.sender, _balance);
}
```

### Whitelisting strategies

The Beefy Keeper can whitelist a strategy address as long as there isn't an active strategy that has funds deployed in the same gauge as the new strategy. An old strategy must be panicked before a new strategy for the same gauge can be tested, so user funds are always protected. The approval for the token ( `_want` ) assigned to the gauge is reset and increased to the max limit for spending by the gauge. The gauge is mapped to the whitelisted strategy and the strategy is allowed access to the GaugeStaker for the specified gauge.

```
// whitelists a strategy address to interact with the Gauge Staker and gives approvals
function whitelistStrategy(address _strategy) external onlyManager {
    IERC20Upgradeable _want = IGaugeStrategy(_strategy).want();
    address _gauge = IGaugeStrategy(_strategy).gauge();
    require(IGauge(_gauge).balanceOf(address(this)) == 0, '!inactive');
    _want.safeApprove(_gauge, 0);
    _want.safeApprove(_gauge, type(uint256).max);
    whitelistedStrategy[_gauge] = _strategy;
}
```

### Upgrading strategies

A new strategy for a gauge with an existing strategy can be proposed once it has been fully tested. `proposeStrategy()` should be called on the GaugeStaker before `upgradeStrat()` on the vault so the switch will succeed. The new strategy must have the same gauge as the previous strategy. `upgradeStrategy()` is only called in `retireStrat()` on the previous strategy, so is controlled indirectly by the vault owner through upgrading the strategy address on the vault.

```
// prepare a strategy to be retired and replaced with another
function proposeStrategy(address _oldStrategy, address _newStrategy) external onlyManager {
    require(IGaugeStrategy(_oldStrategy).gauge() == IGaugeStrategy(_newStrategy).gauge());
    replacementStrategy[_oldStrategy] = _newStrategy;
}

// switch over whitelist from one strategy to another for a gauge
function upgradeStrategy(address _gauge) external onlyWhitelist(_gauge) {
    whitelistedStrategy[_gauge] = replacementStrategy[msg.sender];
}
```

## Contracts

- binSPIRIT/GaugeStaker: 0x44e314190D9E4cE6d4C0903459204F8E21ff940A
- Reward Pool: 0xFAE44b30F6F9BbD44E6B7687471dd73D71FaBDC6

Critical functions are always managed via multi-sig transactions and timelocks. No funds are stored directly on the GaugeStaker and only the active whitelisted strategy can interact with funds stored in a gauge.

# DelegateRegistry Contract

Last Update: February 2023

The [DelegateRegistry contract](#) is a governance smart contract developed by Gnosis and used by Snapshot Labs to facilitate vote delegation in Snapshot-based governance spaces. Users can authorise another user to vote on their behalf using their voting power, by delegating to them on the BNB chain. Users can also remove their delegations at any time.

Through this mechanism, trusted voices in the community can leverage their support with a small amount of effort on the part of their supporters. It also allows those short on time to ensure that their voting power is participating in governance, without requiring them to engage with every proposal that arises.

## Contract Mapping

The DelegateRegistry contract is first and foremost a repository of information on existing delegations, which are stored in the "delegation" mapping in the contract:

```
// The first key is the delegator and the second key a id.
// The value is the address of the delegate

mapping (address => mapping (bytes32 => address)) public delegation;
```

Effectively, each delegation is stored by reference to the delegator's address (i.e. the user that is delegated their voting power), which is then mapped to the relevant Snapshot space ID (stored as a bytes32 value) and the delegate's address (i.e. the user that voting power is delegated to).

For full details of all delegations managed by Snapshot Labs' DelegateRegistry contract, you can explore the Snapshot Subgraph [here](#).

## Contract Events

The DelegateRegistry contract emits two possible events in its ordinary operations.

### SetDelegate

Signifies that the contract's `#setDelegate()` function has successfully been called, and consequently the caller has selected a new delegate.

```
// Using these events it is possible to process the events to build up reverse lookups.
// The indices allow it to be very partial about how to build this lookup (e.g. only for a specific delegate).

event SetDelegate(address indexed delegator, bytes32 indexed id, address indexed delegate);
```

### ClearDelegate

Signifies that one of the below `# Contract Functions` has successfully been called, and consequently the former delegate has been cleared for the caller.

```
// Using these events it is possible to process the events to build up reverse lookups.
// The indices allow it to be very partial about how to build this lookup (e.g. only for a specific delegate).

event ClearDelegate(address indexed delegator, bytes32 indexed id, address indexed delegate);
```

## Contract Functions

The functionality behind the DelegateRegistry contract is very straightforward, and consists of just two functions to set and remove delegates.

### setDelegate()

To set a delegate, the contract requires two inputs: the relevant Snapshot space ID (stored as a bytes32 value) and the delegate's address (i.e. the user that voting power is delegated to). It then tests the inputs against a few requirements (e.g. the user is not delegating to itself, to a null address or to its current delegate) before executing the call.

```
/// @dev Sets a delegate for the msg.sender and a specific id.
///      The combination of msg.sender and the id can be seen as a unique key.
/// @param id Id for which the delegate should be set
/// @param delegate Address of the delegate

function setDelegate(bytes32 id, address delegate) public {
    require (delegate != msg.sender, "Can't delegate to self");
    require (delegate != address(0), "Can't delegate to 0x0");
    address currentDelegate = delegation[msg.sender][id];
    require (delegate != currentDelegate, "Already delegated to this address");

    // Update delegation mapping
    delegation[msg.sender][id] = delegate;

    if (currentDelegate != address(0)) {
        emit ClearDelegate(msg.sender, id, currentDelegate);
    }

    emit SetDelegate(msg.sender, id, delegate);
}
```

Where the call is successful, the function will update the delegation mapping to the new delegate address. It then tests whether the user had previously delegated to another user, and if so it will then emit the `# ClearDelegate` event to signify that the old delegate has been removed. Finally it will emit the `# SetDelegate` event to signify that the new delegate has been added.

Please note that the function does not also require the contract to use the `# clearDelegate()` function, which is only used to remove an existing delegate.

### clearDelegate()

To clear the current delegate, the contract requires one input, which is the relevant Snapshot space ID (stored as a bytes32 value). It then tests to ensure that a delegate has in fact been set before executing the call.

```
/// @dev Clears a delegate for the msg.sender and a specific id.
///      The combination of msg.sender and the id can be seen as a unique key.
/// @param id Id for which the delegate should be set

function clearDelegate(bytes32 id) public {
    address currentDelegate = delegation[msg.sender][id];
    require (currentDelegate != address(0), "No delegate set");

    // update delegation mapping
    delegation[msg.sender][id] = address(0);

    emit ClearDelegate(msg.sender, id, currentDelegate);
}
```

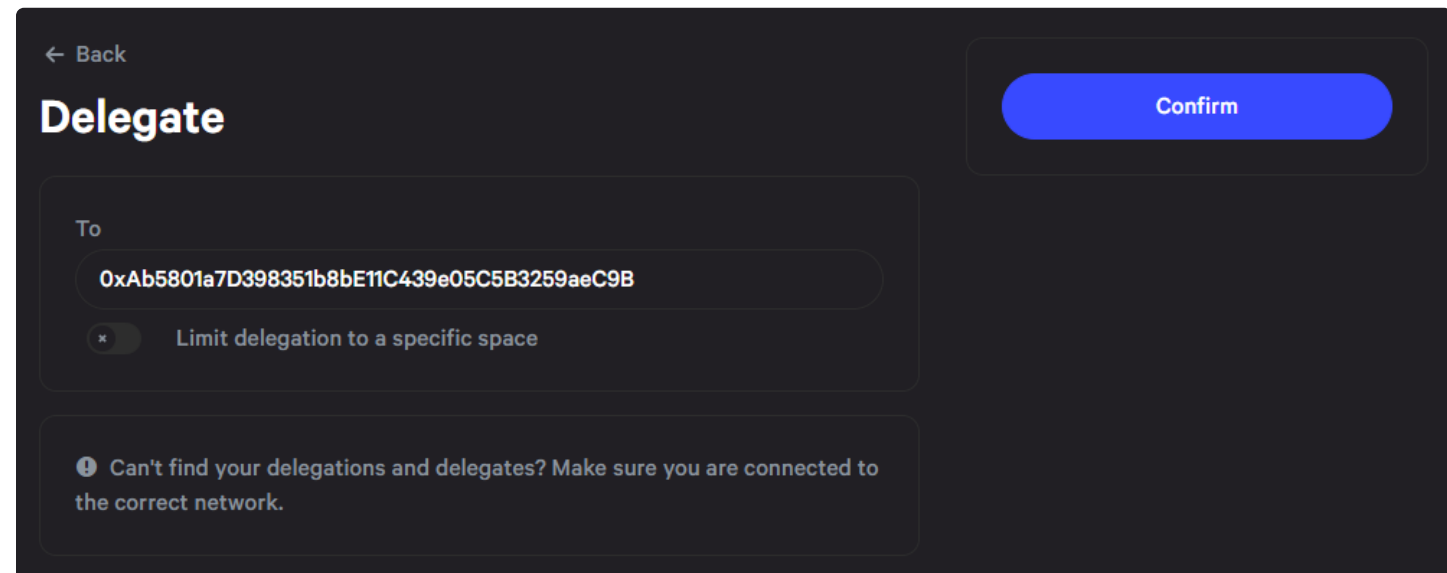
Where the call is successful, the function will update the delegation mapping to the null address (i.e. signifying that the user has not delegated their voting power), and then emits the `# ClearDelegate` event to signify that the old delegate has been removed.

## Delegation Walkthrough

There are two main methods that users can adopt to interact with the DelegateRegistry contract: either interacting through the Snapshot interface or by interacting directly with the contract (e.g. through a relevant block explorer). Below are brief walkthroughs for each methods.

### Through the Snapshot Interface

- Go to the [Snapshot interface delegation page](#).
- Connect your wallet the site, so that it can detect your address and so you can call the `#setDelegate()` function.
- Make sure you're connected to the BNB chain on the BSC network with your wallet. **No other chain will work.**
- Input the address or ENS name of the user you want to delegate your voting power to.
- If you want to, you can select "limit delegation to a specific space" to confine your delegation to only the Beefy Snapshot space. To do so, input the following space: `beefydao.eth`.
- Click "Confirm" to initiate the transaction in your wallet. As this is a write transaction (i.e. submitting information for storage on the blockchain), you will have to pay a small amount of gas to facilitate the transaction.

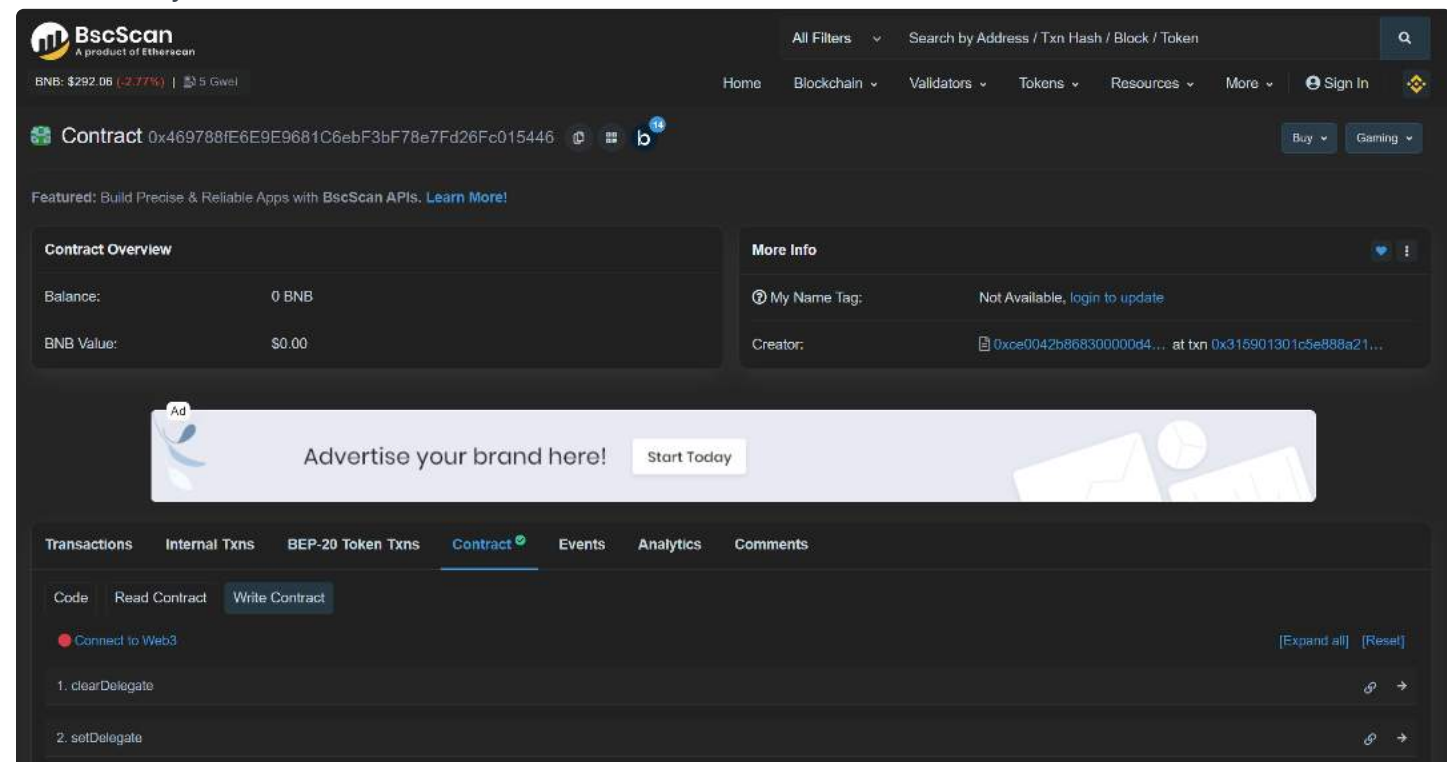


The Snapshot interface's delegation page provides a clean and simple way to delegate your voting power.

- Once the transaction goes through, you will have successfully delegated to the user address that you provided across all chains that our BIFI token is deployed to.

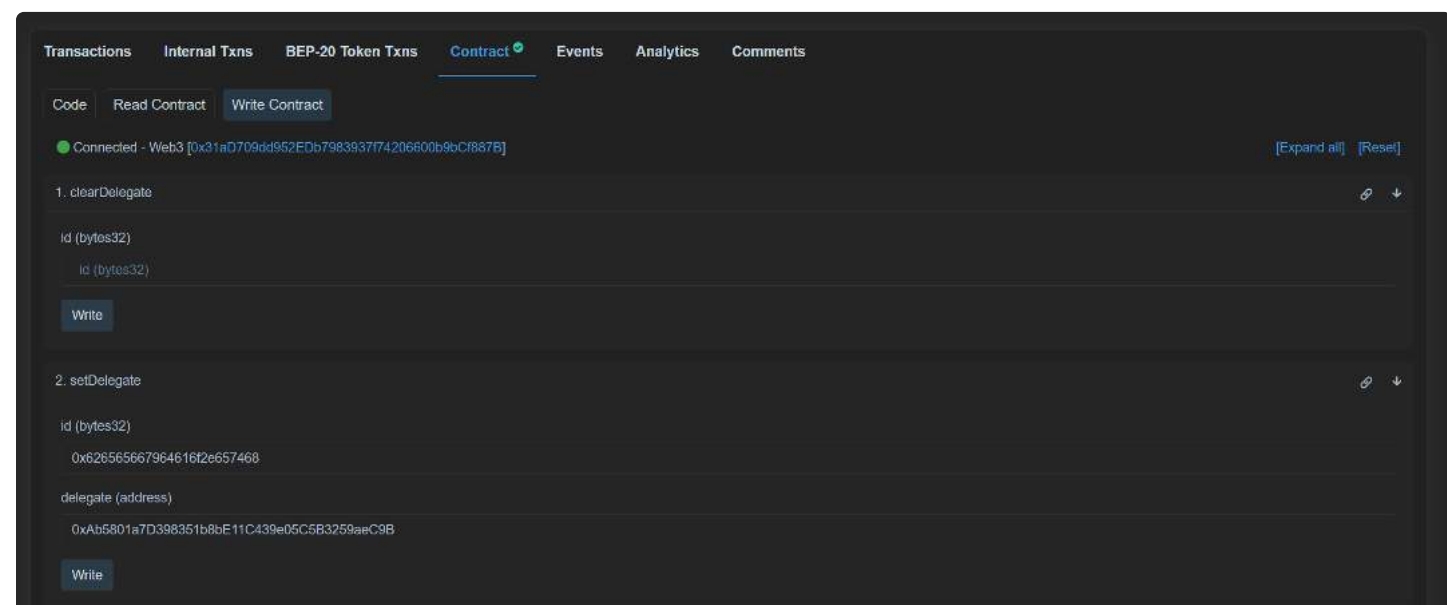
### Through a Block Explorer

- Go to the DelegateRegistry contract address on [BscScan](#).
- Navigate to the "Write Contract" subtab on the "Contract" tab on the contract page (as below).
- Select the "Connect to Web3" button to connect your wallet and interact with the contract.
- Make sure your wallet is set to the BNB chain on the BSC network. **No other chain will work.**



The BscScan interface will provide you with full access to all delegation methods, events and transactions, though they are clearly less user friendly for most users.

- Scroll down to the `# setDelegate()` function, and input the required details, those being:
  - id (bytes32) - the address of the Snapshot space for which you are delegating (i.e. the Beefy Space ID: `0x626565667964616f2e657468`).
  - delegate (address) - the address of the user you want to delegate your voting power to.
- Click "Write" to initiate the transaction in your wallet. As this is a write transaction (i.e. submitting information for storage on the blockchain), you will have to pay a small amount of gas to facilitate the transaction.



The functions on each block explorer reflect those detailed in the `# Contract Functions` section above.

- Once the transaction goes through, you will have successfully delegated to the user address that you provided across all chains that our BIFI token is deployed to.

## Contracts

The DelegateRegistry contract and Beefy snapshot space are deployed at the following addresses:

- DelegateRegistry - `0x469788fE6E9E9681C6ebF3bF78e7Fd26F015446`.
- Beefy Space ID - `0x626565667964616f2e657468` or `beefydao.eth`.

## More Information

For more details of how vote delegation works for Beefy's governance, see the [Governance](#) page (specifically `# How do I delegate my vote?` and `# How do I become a delegate?`). You can also find the current maintained list of vote delegates in this [Google sheet](#).

For full details of all delegations managed by Snapshot Labs' DelegateRegistry contract, you can explore the Snapshot Subgraph [here](#). Further information is available in Snapshot's [documentation](#).

# Beefy API

Last Update: August 2023

This page provides further details about the functionality and operation of Beefy's REST API, which powers our web application, dashboard and pages on third party sites.

You can access the API at <https://api.beefy.finance/>, and the public repository is available on [the Beefy GitHub](#). The API is maintained under the MIT license, with further details available on the [GitHub repo](#).

## Endpoints

Our API offers a range of public endpoints covering the full selection of data required for our web app, backend and dashboard, as well as some third party services.

### Beefy Vault Endpoints

Endpoints developed for use by [the Beefy Application](#) to display vaults' live characteristics and performance to our users.

> GET /vaults

> GET /apy

> GET /apy/breakdown

> GET /tvl

> GET /fees

### External Asset Endpoints

Endpoints developed for use by [the Beefy Application](#) to display information about the assets underlying our Beefy vaults to our users.

> GET /lps

> GET /lps/breakdown

> GET /tokens

### Other Beefy App Endpoints

Endpoints developed for use by [the Beefy Application](#) to display other information not relating to individual vaults.

> GET /config

> GET /boosts

> GET /bifibuyback

### Dashboard Endpoints

Endpoints developed for [the Beefy Dashboard](#) to display overarching information about the protocol.

**i** Please note that the Dashboard site and the /earnings endpoint are both no longer actively maintained by Beefy. Though the Dashboard site remains live, it does not reflect every vault that has been implemented and every chain that Beefy has deployed to.

### Databarn Endpoints

Endpoints related to databarn, our historical data indexer. These endpoints are rate limited to maximum 5 requests per seconds. You may also want to try the [interactive swagger ui](#) for these endpoints.

> GET /api/v1/beefy/timeline  
Fetches the investor timeline

> GET /api/v1/beefy/product/{chain}  
Fetches all product configurations for a chain

> GET /api/v1/beefy/product/{chain}/{contract\_address}  
Fetch a specific product configuration

> GET /api/v1/beefy/product/{chain}/{contract\_address}/tvl  
Get tvl snapshots in use for a specific product.

> GET /api/v1/price/  
Get a quick price time series for a given product and time bucket

> GET /api/v1/price/raw  
Get a raw historical time series for a given time range.

### Third Party Endpoints

Endpoints required or utilised by third party platforms to display information about Beefy or its products on their sites.

> GET /cmc

> GET /supply

## Subgraph

Please note that Beefy does not currently operate or maintain any subgraphs for our protocol.

Our friends at Messari have developed a subgraph for our presence on BSC chain, which is available at <https://api.thegraph.com/subgraphs/name/messari/beefy-finance-bsc>, and on The Graph's [website](#) (ID: QmFtMEjjik9FSZdqAmp2DkNFG4M9TK4Go8uyCUj8EVxY6). Beefy has had no role in the development or maintenance of this subgraph. Please direct any questions or requests for assistance in relation to this subgraph directly to Messari.

## Other Data

For any readers seeking further information about Beefy and our protocol's performance, please head to the [#beefy-data](#) channel in our [Discord server](#). We'll be happy to answer any questions you may have there and you're welcome to inspect the history of our discussions in that channel for further background into our data work.

You'll also find in that channel a collection of weekly reports on the breakdown of our \$BIFI token across our different chains and vaults, which are lovingly prepared by core contributor EPETE.

# Code Repositories



## Frontends

- Beefy Vaults
  - Site: <https://app.beefy.finance/>
  - Repo: <https://github.com/beefyfinance/beefy-app>
- Beefy Landing
  - Site: <https://beefy.finance/>
  - Repo: <https://github.com/beefyfinance/beefy-landing>
- Beefy Governance
  - Site: <https://gov.beefy.finance/staking>
  - Repo: <https://github.com/beefyfinance/beefy-gov>
- Beefy Vote
  - Site: <https://vote.beefy.finance/#/>
  - Repo: <https://github.com/beefyfinance/beefy-vote>
- Beefy Dashboard
  - Site: <https://dashboard.beefy.finance/>
  - Repo: <https://github.com/beefyfinance/beefy-dashboard>

## Backend

- Beefy API
  - Site: <https://api.beefy.finance/>
  - Repo: <https://github.com/beefyfinance/beefy-api>
  - Further details available on the [📄 Beefy API](#) page.

## Smart Contracts

- [BIFI Token](#)
- BIFI Distribution:
  - [Pools](#)
  - [Team Timelocks](#)
  - [Misc.](#)