Ultimate Guide for Launching Currencies on Verus

Launching currencies on Verus, and any other PBaaS-chain (Public Blockchains as a Service), is better, faster, cheaper and more secure than any EVM-like protocol out there. This article explains why, gives an in-depth view of all the options and parameters, and teaches how to launch these currencies. Welcome to Verus, a fundamentally different protocol, and the future of currency launches and value creation.

Max Theyse · Follow
Published in Verus Coin · 16 min read · Jun 16, 2023

532

What are Currencies?

involved. It is just creating a transaction, and then sending that transaction to the blockchain making the currency a reality. Anyone can do it.

Better, Faster, Cheaper and More Secure

Why is it smarter to launch currencies on the Verus Protocol? The answer is simple, because all launched currencies are part of the consensus mechanism. On EVM-like and account-based protocols (Ethereum, BNB Chain, Polkadot, Cosmos, Avalanche) currencies are not part of consensus and thus easily exploitable by smart contract bugs and malicious developers. With Verus, a UTXO model, all currencies are accounted for, and verified by the worldwide miners and stakers of the protocol, making it as secure as sending BTC on Bitcoin. To make it simple — all features on the Verus Protocol are on the first layer, the consensus layer.

> Verus fundamentally revolutionizes the preexisting cryptocurrency landscape

Also, where other protocols need L2 solutions which are mostly centralized, Verus includes self-sovereign identities, namespaces, privacy, DeFi, bridges, and currency and blockchain launches on the first layer. An innovation that fundamentally revolutionizes the preexisting cryptocurrency landscape.

Launching currencies with Verus is also much faster and cheaper than with the alternatives. The magic here is that there is no coding involved. Anyone can launch a currency by just choosing the options and parameters they would like. It is as easy as typing in a command and the currency is ready to be used by the world within minutes. This process does not involve expensive Solidity developers or security audits, saving a lot of development costs.

Another big advantage is being able to export launched currencies to Ethereum as ERC-20. These currencies can then partake in the Ethereum ecosystem, yet have better security and cheaper DeFi transactions than when they are created on Ethereum itself. Each ERC-20 token can also be bridged to Verus. It's all super easy thanks to the non-custodial Verus-Ethereum Bridge.

This is not all some fantasy, this is all functional (the Verus-Ethereum Bridge is in testing), right now, on mainnet, on the Verus Protocol. As you now know, Verus is built fundamentally differently than any other protocol out there, and ushers in a new era for brands, communities, businesses, entrepreneurs and organizations to create value in a better and more secure way than ever before.

The next chapter explains the options to choose from when launching currencies.

Options to Launch Currencies

For ease of explaining, all PBaaS-chain launch options are omitted. Good to know — with Verus anyone can also launch fully interoperable, independent and customizable blockchains with all the same L1 features as the Verus blockchain. The focus of this article is on launching currencies only, on Verus or **any other PBaaS-chain**.

To launch a currency on Verus (and any other PBaaS-chain) a namespace is needed. That namespace is a VerusID. The namespace is the name of the currency (e.g. `MyBrand@`). That namespace can also have subIDs (e.g. `product.MyBrand@`, `user.MyBrand@`, more on this later).

On Verus a VerusID registration costs between 20 and 100 VRSC, and the launch of a currency costs 200 VRSC. These costs are always paid in the chain's native coin (to the worldwide miners and stakers). On other PBaaS-chains these costs can differ since the chain launcher can define its own costs for VerusID registrations and currency launches.

To define a currency, choose options. Then combine these options to whatever the currency needs to be. Just add the numbers together in the currency definition. The currency options are listed below.

- `options:1` — **The currency has reserves**, and can be converted to and from the reserves (option 32 needs to be added). Can have one currency as its reserves, or multiple with up to 10 currencies. Let's call it a "Basket currency" — a currency with a basket of reserves. Such a currency can be launched centralized or decentralized.
- `options:2` — **Only the controlling VerusID** (the namespace of the currency, the rootID) can create subIDs.
- `options:8` — **Referrals and discounts are enabled** for subID creation.
- `options:16` — **Referrals are required** for subID creation.
- `options:32` — The currency is a **simple token currency without any reserves.** Such a currency can be launched centralized or decentralized. This option is also used for Ethereum ERC-20 mapped tokens.

- `options:2048` — A single satoshi (0.00000001) NFT token is created & has tokenized control of the root VerusID. Which means you can send the single satoshi token to other addresses and then they have control of the root VerusID.

Let's give some examples of combined options:

- `"options":33` — this launches a **basket currency**. It is options 1 + 32 combined.
- `"options":2080` — this launches a **single satoshi token** that has control of the root VerusID. It is options 32 + 2048 combined.
- `"options":35` — this launches a **basket currency,** and **only** the rootID can create subIDs. It is options 1 + 2 + 32 combined.

And so on. More information on basket currencies, simple token currencies and subIDs in the next chapter.

Two Types of Currencies

There are two types of currencies that can be launched with the Verus Protocol. Basket currencies and simple token currencies. Both can be issued decentralized, or centralized (has minting and burning capabilities.)

Basket Currencies

A simple example of a basket currency containing two reserve currencies.

Basket currencies are magical currencies that have reserves. Have a look at the simplified image. If anyone has currency X or Y, they can convert to the basket currency, or convert from reserve to reserve. If anyone has the basket currency, they can go to currency X or Y. A basket currency can have 1 and up to 10 reserve currencies.

The basket currency supply is dynamic, depending on how much is converted to the basket currency (supply minted), or back to its reserve(s) (supply burned).

A basket currency can be 100% backed by its reserves, 5%, or anything in between. This is called the reserve ratio, or the weight. The lower the reserve ratio, the more volatile the currency is when people are converting into or out of the basket currency. The value of the basket currency is directly linked to what is in the reserves and what the reserve ratio is.

When a centralized version of this currency is created, the owner of the rootID can mint currencies into existence, while automatically lowering the reserve ratio. Or they can burn currencies and automatically raise the reserve ratio. Anyone can also just burn the currency at will without raising the reserve ratio.

The conversion fees are incredibly low, 0.025% when converting to and from a basket currency, and 0.05% when converting from reserve to reserve currency. These fees go directly to the worldwide miners and stakers of the protocol, and/or they are accrued into the reserves making the basket currency worth more.

Because all currency conversions are solved simultaneously inside a block, giving **all participants the same price**, the protocol is MEV-free. The protocol doesn't have any of the problems EVM-like account-based systems have. Verus DeFi is fair, cheap and has no rent-seekers.

Every(!) currency on the Verus network (also mapped ERC-20s!), can be used as reserves for basket currencies. As you might start to understand now — basket currencies are unique currencies that can not be found anywhere else and offer an enormous amount of opportunities for value creation. And anyone can launch them without needing any programming knowledge!

Simple Token Currencies

Simple token currencies are just currencies without any reserves. They are not as exciting as the basket currencies, yet still offer much value. With all the parameters that can be added, subIDs created and decentralized crowdfund mechanism, these currencies can support a lot of use cases that are difficult to do with alternative protocols. A small example of a centralized currency here.

The supply of this type of currency is static when it's a decentralized version. When it's a centralized version, the owner of the rootID can mint tokens into existence, and anyone can burn them.

This option is also used to create currencies that are mapped to Ethereum ERC-20s. Which means you can send those ERC-20s over to Verus, or from Verus to the ERC-20. This is made possible with the non-custodial Verus-Ethereum Bridge. You can read more about it here.

And of course, a simple token currency can be one of the reserves in a basket currency.

What are SubIDs

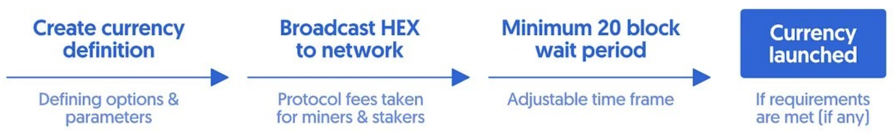
SubIDs are powerful objects on the blockchain. They can be registered in an unlimited amount under any created currency (basket and simple token currency). A few examples are `product.MyBrand@` or `user.MyBrand@`. The name `MyBrand@` is a VerusID (a namespace) that is converted into a currency. The subIDs are 'product' and 'user'.

SubIDs can be bound digitally to many things. They can be bound to people and products. They can be bound to an unlimited amount of content, data, and provable information, both public and private. Including provable contracts and rights that can be bound to ownership of the subID itself.

Another powerful feature is that it has read, write and delete functions in what is called the "content multimap". All kinds of information can be read and processed in the mempool of the blockchain.

It doesn't stop there, it's also a friendly name blockchain address that can receive, send, hold and stake funds, and can be used to login to services, password-free. Read more about VerusID here.

Parameters to Launch Currencies


Create currency definition
Defining options & parameters
Broadcast HEX to network
Protocol fees taken for miners & stakers
Minimum 20 block wait period
Adjustable time frame
Currency launched
If requirements are met (if any)

This is all anyone needs to do to launch powerful currencies. Zero coding!

Next up are the parameters. Choose the parameters wisely to launch a currency that suits any need. Not all parameters are needed or combinable. There are many to go through, so let's start.

“proofprotocol”

This parameter defines, among others, if the currency is centralized or decentralized. You can choose 1,2 or 3.

`1` is default, which launches a decentralized currency (no need to include this parameter when defining such a currency). When subIDs are created with this option, the registration fees are burned.

`2` is for a centralized currency. If it’s a basket currency the rootID can mint and burn supply while automatically lowering and raising the reserve ratio (also anyone can burn supply without lowering the reserve ratio). Or when it’s a simple token currency they can just mint and burn supply. The subID registration fees go to the rootID.

`3` is for Ethereum ERC-20 mapped tokens. Read more here

Example currency:

```
definecurrency '{
  "name":"MyBrand",
  "options":32,
  "proofprotocol":2,
  "preallocations":[{"Klaus@":100}]
}'
```

🔍 *A simple token currency called MyBrand, centralized (the controller of the rootID can mint and burn), and has a preallocation of 100 tokens to Klaus@.*

“currencies”

Here you put the names of the currencies (or just one — it must have VRSC when launched on Verus) that will be in the reserves when it’s a basket currency (`options:33`).

Or when it’s a simple token currency (`options:32`), what people convert during the preconversion timeframe will go to the rootID of the currency, as a funding mechanism. In the case of a simple token currency, combine it with `"conversions"` to determine the preconversion price.

Example currency:

```
definecurrency '{
  "name":"CommunityX",
  "options":33,
  "currencies":["vrsctest","MyBrand"],
  "minpreconversion":[10,50],
  "initialsupply":100
}'
```

🔍 *A basket currency called CommunityX. It needs to get a minimum of 10 VRSCTEST and 50 MyBrand into its reserves within the preconversion time frame to be launched. The initial supply of 100 CommunityX will be distributed among the preconverters.*

“conversions”

Use this parameter when launching a simple token currency. Together with `"currencies"`, it can be used as a funding mechanism for the rootID. It’s for the preconversion price. So when doing `"conversions":[0.1]`, it means that for every VRSC the preconverter receives 10 CURRENCY after launch. The converted VRSC goes into the rootID.

People can preconvert to this currency within the preconversion time frame. Define a `"startblock"`, or let the default and minimum time frame play out, which is 20 blocks.

Example currency:

```
definecurrency '{
  "name":"CoolBrand",
  "options":32,
  "currencies":["vrsctest"],
  "conversions":[0.1],
  "minpreconversion":[1000]
}'
```

🔍 *This simple token currency is called CoolBrand. During the preconversion time frame people need to convert 1000 VRSCTEST to the rootID. In exchange for that they receive 10.000 CoolBrand. If this minimum amount is not met, the currency will not launch, and everyone who did a preconvert will get their funds back.*

"minpreconversion"

Use this parameter to set a minimum amount of preconversions. The minimum amount of preconversions needs to be met or the currency will not launch and everyone gets their conversions returned, minus the transaction and conversion fees. It works both with basket currencies and simple token currencies.

There is a 0.025% fee taken when preconverting. Take this into consideration when trying to meet the minimum amount of preconversions.

Example currency:

```
definecurrency '{
  "name":"CompanyX",
  "options":32,
  "currencies":["vrsctest"],
  "conversions":[2],
  "minpreconversion":[500]
}'
```

🔍 *This simple token currency is called CompanyX. During the preconversion time frame people need to convert 500 VRSCTEST to the rootID. In exchange for that they receive 250 CompanyX. If this minimum amount is not met, the currency will not launch, and everyone who did a preconvert will get their funds back.*

"maxpreconversions"

Use this parameter to set a maximum amount of preconversions. During the preconversion time frame the amount set can not be exceeded. Everything above this amount will be automatically refunded after the currency is launched.

Example currency:

```
definecurrency '{
  "name":"CoinCommunity",
  "options":33,
  "currencies":["vrsctest"],
  "maxpreconversion":[100],
  "initialsupply":100
}'
```

🔍 *This is a basket currency called CoinCommunity. During the preconversion time frame people can convert VRSCTEST into its reserves for 100 CoinCommunity in return. During the preconversion time frame there can not be more than 100 VRSCTEST converted into its reserves. Whatever is preconverted more will be returned.*

"initialcontributions"

The rootID can contribute some or all of the minimum preconversions directly as part of the currency definition. Use this parameter to make an initial contribution to either the reserves when it's a basket currency (`options:33`), or to the rootID when it's a simple token currency (`options:32`).

The funds to initially contribute need to be in the rootID when defining the currency. After the preconversion time frame is over and the currency launched, the rootID has received an amount of the launched currency.

Example currency:

```
definecurrency '{
  "name":"CommunityBasket",
  "options":33,
  "currencies":["vrsctest","CoinCommunity"],
  "initialcontributions":[10,200],
  "initialsupply":100,
  "preallocations":[{"Jane@":100},{"John@":50}]
}'
```

🔍 *This is a basket currency called CommunityBasket. The launcher of the currency wanted to make initial contributions to its reserves. At the moment of broadcasting the currency to the network, there needed to be 210 VRSCTEST and 200 CoinCommunity in the rootID. The initial supply of 100 went to the rootID (if there weren't any more preconverters). At the same time of the launch, 100 CoinCommunity was minted into Jane@ and 50 into John@, this lowered the reserve ratio of the currency.*

"initialsupply"

A required parameter for basket currencies (`options:33`). Does not work with simple token currencies. This is the initial supply during the preconversion time frame, before the currency is launched. People preconverting into the reserves receive from this initial supply.

Immediately after the currency is launched, the supply can be larger due to `"preallocations"`.

Example currency:

```
definecurrency '{
  "name":"SocialBrand",
  "options":33, "currencies":["vrsctest"],
  "initialsupply":500,
  "preallocations":[{"Max@":1000}]
}'
```

🔍 *This is a basket currency called SocialBrand. People can preconvert VRSCTEST into its reserves and in return they get 500 SocialBrand distributed among them. Immediately after launch Max@ receives 1000 SocialBrand, lowering the reserve ratio of the currency.*

"preallocations"

Use this parameter to receive a chosen amount of funds after the preconversion time frame has passed and the currency is launched. Funds can be directed to VerusIDs. Works with simple token currencies and basket currencies.

When using this parameter with basket currencies, after the preconversion time frame has passed and the currency is launched, the reserve ratio is lowered. This is because new currency has been minted after the initial supply (`"initialsupply"`) is distributed.

"prelaunchcarveout"

Only works with basket currencies (`options:33`). Use this to redirect a percentage of preconverted reserves to the rootID.

After the preconversion time frame has passed and the currency is launched, a percentage of the reserves is taken and redirected to the rootID. This lowers the reserve ratio, making the currency more volatile.

Example currency:

```
definecurrency '{
  "name":"BusinessBrand",
  "options":33,
  "currencies":["vrsctest"],
  "initialsupply":100,
  "prelaunchcarveout":0.1
}'
```

🔍 *This is a basket currency called BusinessBrand. People can preconvert VRSCTEST into its reserves in return for 100 BusinessBrand distributed among them. When the currency is launched, 10% VRSCTEST is taken out of the reserves, into the rootID. This lowers the reserve ratio by 10%.*

"prelaunchdiscount"

Only works with basket currencies (`options:33`). Use this to give people a discount during the preconversion time frame. After the preconversion time frame and the currency is launched, the conversion price will be higher, depending on what percentage the discount was.

When using this parameter, after the currency is launched, the reserve ratio will be lowered by the discounted percentage.

Example currency:

```
definecurrency '{
  "name":"DiscountBrand",
  "options":33,
  "currencies":["vrsctest"],
  "initialsupply":100,
  "prelaunchdiscount":0.5
}'
```

🔍 *This is a basket currency called DiscountBrand. People can preconvert VRSCTEST into its reserves in return for 100 DiscountBrand distributed among them. Immediately after the launch of the currency, when people want to convert, the price is 50% higher. Also, the reserve ratio is 50% lower because of the prelaunchdiscount.*

"weights"

Only works with basket currencies (`options:33`). Use this to change the respective weights of the reserves in a basket currency. The total of all weights must equal 1. With a minimum of 0.1, since there can't be more than 10 reserve currencies in a basket currency.

Example currency:

```
definecurrency '{
  "name":"MyBusiness",
  "options":33,
  "currencies":["vrsctest","BusinessBrand","DiscountBrand"],
  "initialsupply":100,
  "weights":[0.5,0.25,0.25]
}'
```

🔍 *This is a basket currency called MyBusiness. During the preconversion time frame there are various currencies that can be converted into its reserves. They have different weights to them. 0.5 for VRSCTEST, 0.25 for both BusinessBrand and DiscountBrand.*

"startblock"

Use this parameter to define the block height when the currency is launched. There is a preconversion time frame before the currency is launched. When omitting this parameter it uses a 20 block preconversion time frame before the currency is launched.

The preconversion time frame is always 20 blocks, this can not be less.

“endblock”

Endblock can not be defined on basket currencies. It does nothing. It could be set as a signal to software that might use the basket currency.

It can be set on centralized (`proofprotcol:2`) simple token currencies. When the endblock is reached, it turns the centralized currency into a decentralized one.

“idregistrationfees”

Use this parameter to change the costs of registering subIDs under the rootID. The default registration fee is 100.

When it’s a decentralized currency the fees are burned, when it’s centralized the fees go to the rootID.

To enable discounts when using referrals, add `"options":8` to the currency definition.

“idreferrallevels”

Use this parameter to change the levels of referrals used when registering subIDs. The default is 3 levels.

“nativecurrencyid”

Use this parameter for mapped ERC-20 tokens. The parameter includes the Ethereum contract address for the ERC-20. Read here for more information.

. . .

These were all the parameters to use when defining currencies. There are many, and combining them in the right ways make powerful currencies and tokens. The community is always ready to help when defining and launching currencies for your use cases! **Please go to Discord and ask away.**

The next chapter explains how to exactly launch these currencies.

How to Launch the Currencies

Launching these currencies is quite easy. Launch with the command-line interface, or Verus Desktop for Windows, Linux or macOS. **Download here.**

In the chapter before many examples of currency definitions are given. Let’s use one and see how it is launched.

First of all let’s see the difference between launching with Verus Desktop and the command-line interface. Verus testnet is used for testing purposes.

⚠️ **It is highly recommended to launch currencies on testnet first before going to mainnet!** ⚠️

Verus Desktop: go to “Settings” (cogwheel top-right) -> “Coin Settings”. There you see an input field.

Here you can run commands in Verus Desktop

To use the input field and launch a currency on Verus Desktop (see `run` *before the command):*

```
run definecurrency '{
  "name":"MyBrand",
  "options":32,
  "proofprotocol":2,
  "preallocations":[{"Klaus@":100}]
}'
```

Command-line interface: (this is for testnet, when on mainnet omit `-chain=VRSCTEST` or for any other PBaaS-chain, just edit `VRSCTEST`)

```
./verus -chain=VRSCTEST definecurrency '{
  "name":"MyBrand",
  "options":32,
  "proofprotocol":2,
  "preallocations":[{"Klaus@":100}]
}'
```

Now, after doing these commands in either Verus Desktop or the command-line interface you get a HEX returned. It's a very large array of letters and numbers. We need to use this HEX to broadcast the currency to the blockchain. Only then will the currency really start and the funds taken (distributed to the miners and stakers) for the launch.

Using the HEX

To start the currency use these commands:

Verus Desktop:

```
run sendrawtransaction "HEX"
```

Command-line interface: (this is for testnet, when on mainnet omit `-chain=VRSCTEST` or for any other PBaaS-chain, just edit `VRSCTEST`)

```
./verus -chain=VRSCTEST sendrawtransaction "HEX"
```

After doing these commands the currency has started, the funds from the rootID are taken, and it takes a minimum of 20 blocks to actually launch it (and if all preconversion goals are met).

Currency information

During the preconversion time frame and after the launch you can lookup all kinds of information on the currency.

Verus Desktop:

```
run getcurrency "MyBrand"
```

Command-line interface: (this is for testnet, when on mainnet omit `-chain=VRSCTEST` or for any other PBaaS-chain, just edit `VRSCTEST`)

```
./verus -chain=VRSCTEST getcurrency "MyBrand"
```

Now you know everything to launch currencies! 👉 The worldwide community is happy to help on Discord!

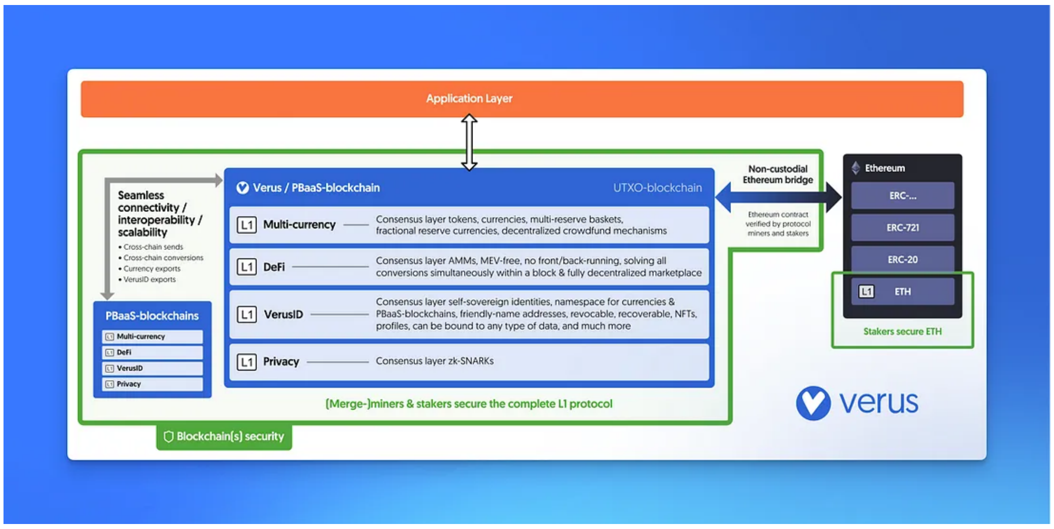
. . .

Start to Build

For all the builders out there, you can start to use the Verus Protocol right now for your Dapps or businesses. Building with Verus is better, faster, cheaper and more secure than anything else out there.

Focus on building the application layer instead of spending giant amounts of time and money on Solidity developers and smart contract audits.

Launch powerful currencies for your brand, business, community and organization **now**.



Application Layer
Seamless connectivity / interoperability / scalability
• Cross-chain sends
• Cross-chain conversions
• Currency exports
• VerusID exports
PBaaS-blockchains
Multi-currency
DeFi
VerusID
Privacy
Verus / PBaaS-blockchain
UTXO-blockchain
L1 Multi-currency
Consensus layer tokens, currencies, multi-reserve baskets, fractional reserve currencies, decentralized crowdfund mechanisms
L1 DeFi
Consensus layer AMMs, MEV-free, no front/back-running, solving all conversions simultaneously within a block & fully decentralized marketplace
L1 VerusID
Consensus layer self-sovereign identities, namespace for currencies & PBaaS-blockchains, friendly-name addresses, revocable, recoverable, NFTs, profiles, can be bound to any type of data, and much more
L1 Privacy
Consensus layer zk-SNARKs
Non-custodial Ethereum bridge
Ethereum contract verified by protocol miners and stakers
Ethereum
ERC-...
ERC-721
ERC-20
L1 ETH
Stakers secure ETH
(Merge-)miners & stakers secure the complete L1 protocol
Blockchain(s) security
verus

Image showing the L1 primitives of Verus, and how you only need to focus on the application layer when building your Dapp or business!

Join the community. Learn about the protocol. Use Verus. Get ahead of the game.

➡️ Join the community on Discord

Follow on Twitter

Go to verus.io

. . .

Special thanks to community member **ejuliano** for putting in a lot of work that got me started on this! 🙏

Blockchain Technology · Blockchain Development · Cryptocurrency Investment · Cryptocurrency · Blockchain

532

Written by Max Theyse

149 Followers · Editor for Verus Coin

Follow

More from Max Theyse and Verus Coin

Max Theyse in Verus Coin

Introducing Pure — The Currency 100% Backed by Verus & Bitcoin

Pure is a decentralized currency fully backed by Verus (VRSC) & Bitcoin (tBTC).

8 min read · Mar 16, 2024

196

Max Theyse in Verus Coin

Verus: Profit Generating Protocol for Miners and Stakers

Future and present Verus miners: take notice. Be ready to maximize profit with your...

4 min read · May 12, 2021

304

Oliver in Verus Coin

How to Start CPU Mining Verus Coin VRSC from Your Computer i...

A Dummies Step by Step Guide to Pool Mining Verus Coin with a CPU!

8 min read · Jan 10, 2019

346 1

Max Theyse

How to preconvert into Bridge.vARRR

Participate in the first ever Verus PBaaS-chain launch — vARRR. Learn how to...

4 min read · Mar 20, 2024

259

See all from Max Theyse

See all from Verus Coin

Recommended from Medium

Max Theyse in Verus Coin

Introducing Pure — The Currency 100% Backed by Verus & Bitcoin

Pure is a decentralized currency fully backed by Verus (VRSC) & Bitcoin (tBTC).

8 min read · Mar 16, 2024

196

Albert Peter in Cryptocurrency Scripts

Top 5 Crypto Gems Predicted to Reach 100x-1000x Gains in 2024

Discover the top 5 crypto gems poised for 100x-1000x gains in 2024. Don't miss out on...

6 min read · Mar 14, 2024

538 13

Lists

Modern Marketing

103 stories · 523 saves

Generative AI Recommended Reading

52 stories · 894 saves

My Kind Of Medium (All-Time Faves)

71 stories · 261 saves

Perzibal

How to mine TAI using TonAi on telegram ? Full guide.

Start Mining on Telegram bot for Free

2 min read · Feb 11, 2024

4

Edge Ruler in Coinmonks

Top 3 Cryptos to Hold until 2025: Like Buying BITCOIN at 10$

In today's crypto landscape, amidst a prolonged bear market, investors find...

· 4 min read · Mar 16, 2024

781 4

Ahmad Rizwan

12 Lucrative Free Mining Apps To Make Some Serious Money in 2024

Discover the Top Cryptocurrency Mining Apps That Can Generate Profits Without...

· 5 min read · Mar 19, 2024

180

Passive Crypto Mining

My Top 15 Passive Crypto Miners of 2024

I've spent 2 weeks compiling the most profitable Crypto Miners for 2024. You can...

11 min read · Jan 12, 2024

2K 29

See more recommendations