How to monitor Starknet wallets

Learn how to build a Starknet wallet monitoring service with Juno. Set up real-time alerts on wallet activities like token transfers, contract interactions, whale trades, and more.

Jelilat Anofiu

Follow

Nethermind.eth

--

Listen

Share

By

[Jelilat Anofiu

](https://twitter.com/tjelailah).

Wallet monitoring is no doubt an important aspect of blockchain interaction. Whether it's for keeping tabs on your transactions, getting notified about strange activities on specific addresses or even tracking whale trades, a robust wallet monitoring service is invaluable.

In this tutorial, I'll guide you through the process of building a wallet monitoring service using Juno, our open-source Starknet node client. This step-by-step guide will cover everything from setting up your node to building the monitoring service itself.

Setting up your Starknet node with Juno

Before diving into the nitty-gritty of building the monitoring service, you need to first set up your Starknet node using the Juno client. The node enables you to access and interact with Starknet data in real-time.

To begin the setup process, check out ourcomprehensive documentation. This will walk you through every step and provide additional insights into the functionalities that Juno offers.

Building your wallet monitoring service

In this section, we'll build a wallet monitoring service that sends the user an email whenever a transaction is executed by their wallet. We'll be using TypeScript, NodeJS, and the Mailjet API.

Step 1: Setting up your project

Start by ensuring you have Node.js and npm installed on your machine. If not, you can download and install them fromhere.

Next, create a new directory for your project and navigate into it:

Then, initialize a new Node.js project with npm:

This will create a package.json

file in your directory.

Step 2: Installing dependencies

We'll use the starknet

package to interact with Starknet and the node-mailjet

package to send emails. Install them with npm:

Step 3: Setting up environment variables

We'll be using environment variables to store sensitive information such as the node URL and Mailjet API keys. Create a new file named .env

in your project directory and add the following line:

Replace your_juno_node_url

, your_mailjet_public_key

, and your_mailjet_secret_key

with your actual Juno node URL and Mailjet API keys.

Step 4: Setting up the Email Notification

We'll use Mailjet to send email notifications whenever a transaction is detected. For that, create a new TypeScript file named email.ts

in your project directory. This file will handle sending emails through Mailjet. Feel free to use any email provider of your choice.

In email.ts

, paste the following code.

Don't forget to replace "wallet-monitor@nethermind.io"

with your actual email address and "user@starknet.com"

with the email address you want to send alerts to. Also, make sure to replace process.env.MJ_APIKEY_PUBLIC!

and process.env.MJ_APIKEY_PRIVATE!

with your actual Mailjet API public and private keys.

Now, whenever mailjetRequest(transactionHash)

is called, an email will be sent to the user with the details of the transaction on Voyager.

In the next step, we'll put it all together and create the logic for the wallet monitoring service in index.ts

.

Step 5: Building the Wallet Monitoring Service

Let's go back to our index.ts

file and import the mailjetRequest

function from email.ts

:

Next, create a function stringToHexFelt

that converts strings to [sn_keccak

](https://docs.starknet.io/documentation/architecture_and_concepts/Hashing/hash-functions/).

Create listenToEvents(lastBlockNumber: number)

that listens for new events from your Starknet node. If the function finds a new event, it sends an email to the user:

Lastly, create a loop that keeps checking for new blocks and listens for new events:

Step 6: Running the Wallet Monitoring Service

To run your wallet monitoring service, compile your TypeScript files to JavaScript using the TypeScript compiler. If you

haven’t already installed TypeScript, you can do so with the following command:

Now, compile your TypeScript files to JavaScript:

Finally, run your service with Node.js:

Alternatively, you can run

Your wallet monitoring service is now running! It will send an email to the specified user whenever a new transaction is detected on the monitored wallet, and print the details of the transaction to the console.

Please note that the code provided in this tutorial is intended for educational purposes and is not suitable for production use.

Want to play a part in Juno’s evolution? Start contributinghere 😊

About us

Nethermind is a team of world-class builders and researchers. We empower enterprises and developers worldwide to access and build upon the decentralized web. Our work touches every part of the web3 ecosystem, from our Nethermind node to fundamental cryptography research and infrastructure for the Starknet ecosystem.

If you’re interested in solving some of blockchain’s most difficult problems, visit our

[job board

](https://nethermind.page.link/careers)!

Disclaimer:

This blogpost has been prepared for the general information and understanding of the readers. The views and strategies described may not be suitable for all readers. No third party should rely on this article in any way, including without limitation as financial, investment, tax, regulatory, legal, or other advice, or interpret this article as any form of recommendation.

This blogpost does not indicate Nethermind’s endorsement of any particular project or team, nor guarantee its security. Readers should use their best judgment and exercise caution where appropriate.

To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this blogpost, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.