Smart Contract Triggering with Chainlink Automation

Joe · Follow
Published in Web3 Writers Guild
5 min read · Nov 14, 2023

You might have properly written a smart contract before and you have been told that smart contracts functions must be triggered either by an EOA (Externally owned account) or another smart contract.

And then for every time, you want to make a change to your contract storage, you have to trigger a function using your EOA or writing another smart contract. If you are coming from the web2 space or you have a slight idea of how the backend system works in web2 space. You will know about cron jobs.

Cron jobs are a powerful tool for automating tasks in Node. js. They allow you to run code at specific intervals, such as every minute, hour, day, week, or month. This can be used to do things like send emails, run backups, or update data.

What if you can implement something like that in your smart contracts, and then you don't have to be the one doing the job of calling functions yourself.

Welcome to Chainlink Automation. Let me take you through a summary of how it works

Chainlink Automation are more like triggers that trigger functions executions in smart contracts and it is categorized into three triggers and links to Chainlink to read more ont hem, and there are as follow;

Time-based trigger: Use a time-based trigger to execute your function according to a time schedule. This feature is also called the Job Scheduler and it is a throwback to the Ethereum Alarm Clock. Time-based trigger contracts don't need to be compatible with the `AutomationCompatibleInterface` contract.

Custom logic trigger: Use a custom logic trigger to provide custom solidity logic that Automation Nodes evaluate (off-chain) to determine when to execute your function on-chain. Your contract must meet the requirements to be compatible with the `AutomationCompatibleInterface` contract. Custom logic examples include checking the balance on a contract, only executing limit orders when their levels are met, any one of our coded examples, and many more.

Log trigger: Use log data as both trigger and input. Your contract must meet the requirements to be compatible with the `AutomationCompatibleInterface` contract.

For this example, I will be showing you an implementation of chainlink Automation using Time-based trigger. I will take you through a simple contract written in solidity.

We will create a contract called automatedPay that allows

```
// SPDX-License-Identifier: UNLICENSED

pragma solidity 0.8.18;

contract AutomatedPay {
    address payee;
    uint unlocked;
    uint lockedTime;
    address owner;

    constructor(address _payee, uint _unlocked) {
       payee = _payee;
        unlocked = _unlocked;
        lockedTime = block.timestamp;
        owner = msg.sender;
    }

    error TimeNotReached();

    function deposit() external payable {
        require(msg.value > 0, "ZeroAmount");
    }

    function withdraw() external {
        require(msg.sender == owner, "Unauthorized");
        if (block.timestamp <= (lockedTime + unlocked)) {
            revert TimeNotReached();
        }
        (bool sent, ) = payee.call{value: address(this).balance}("");
        require(sent, "Failed to send Ether");
    }

    receive() external payable { }

}
```

The smart contract above allows withdrawal of funds after the locked time has been reached. The unlocked time is set at constructor level, the account to pay is also set at constructor level.

Deploy the contract above and save the address, you will needing it. Deposit any amount of ethers into the contract and then any account to receive the ethers when the time elapses.

The withdrawal function can only be call by the account that deployed the contract. Now in the case that we ain't closed accessing our wallet, what then happens, we will then have to wait till the we do.

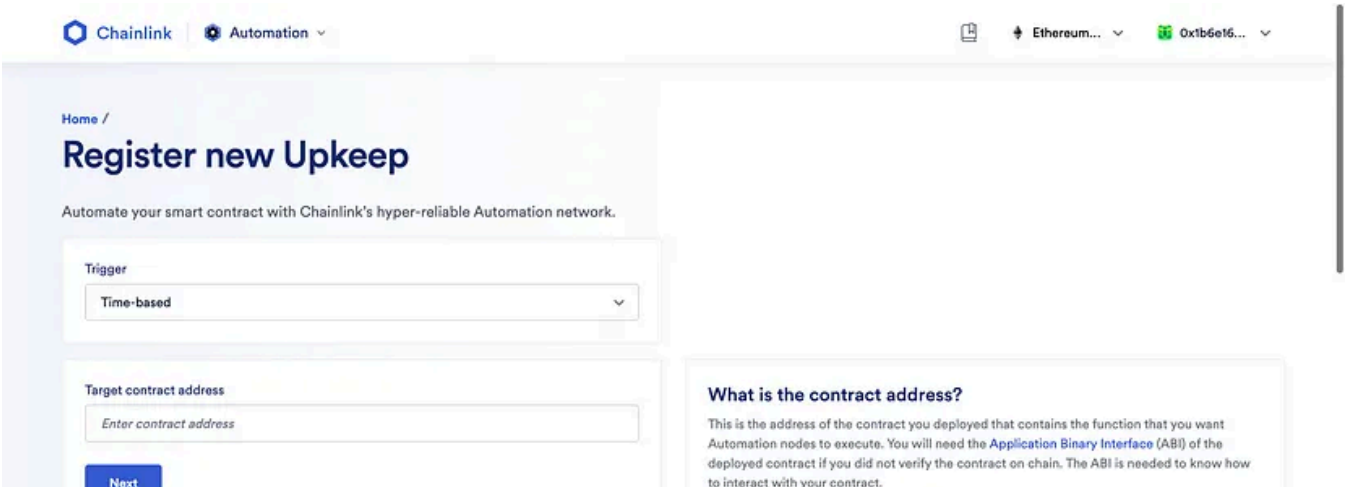
But with chainlink automation, we can automate the contract to pay out the funds when the time elapses.

Let's get to that part.

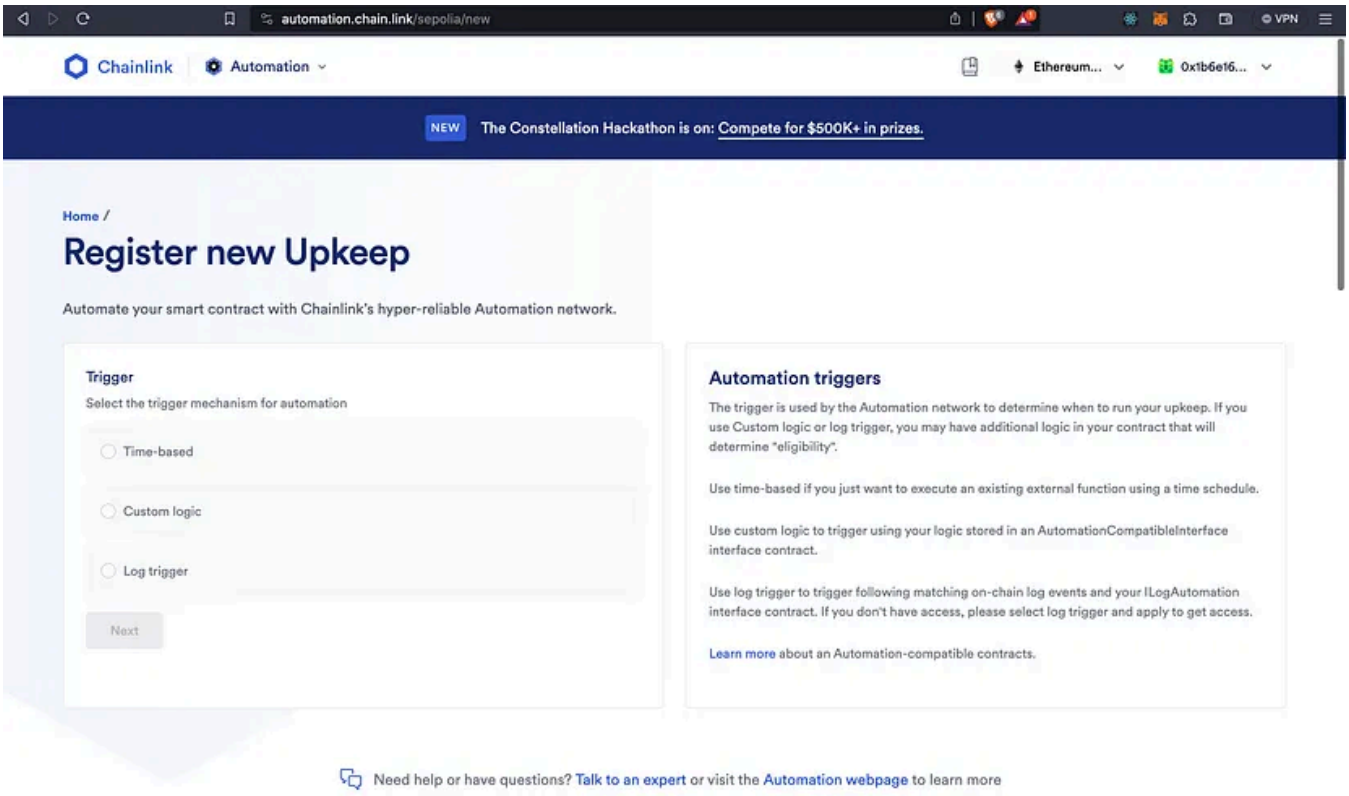
First of all, we will need to set a Time-Based Upkeep to carry out the trigger,

If you are deploying to sepolia, then you will need some sepolia teest net. If its Polygon, then you will need some matic tetsnet.
I deployed my contract to Sepolia, so I will be preceeding with sepolia.

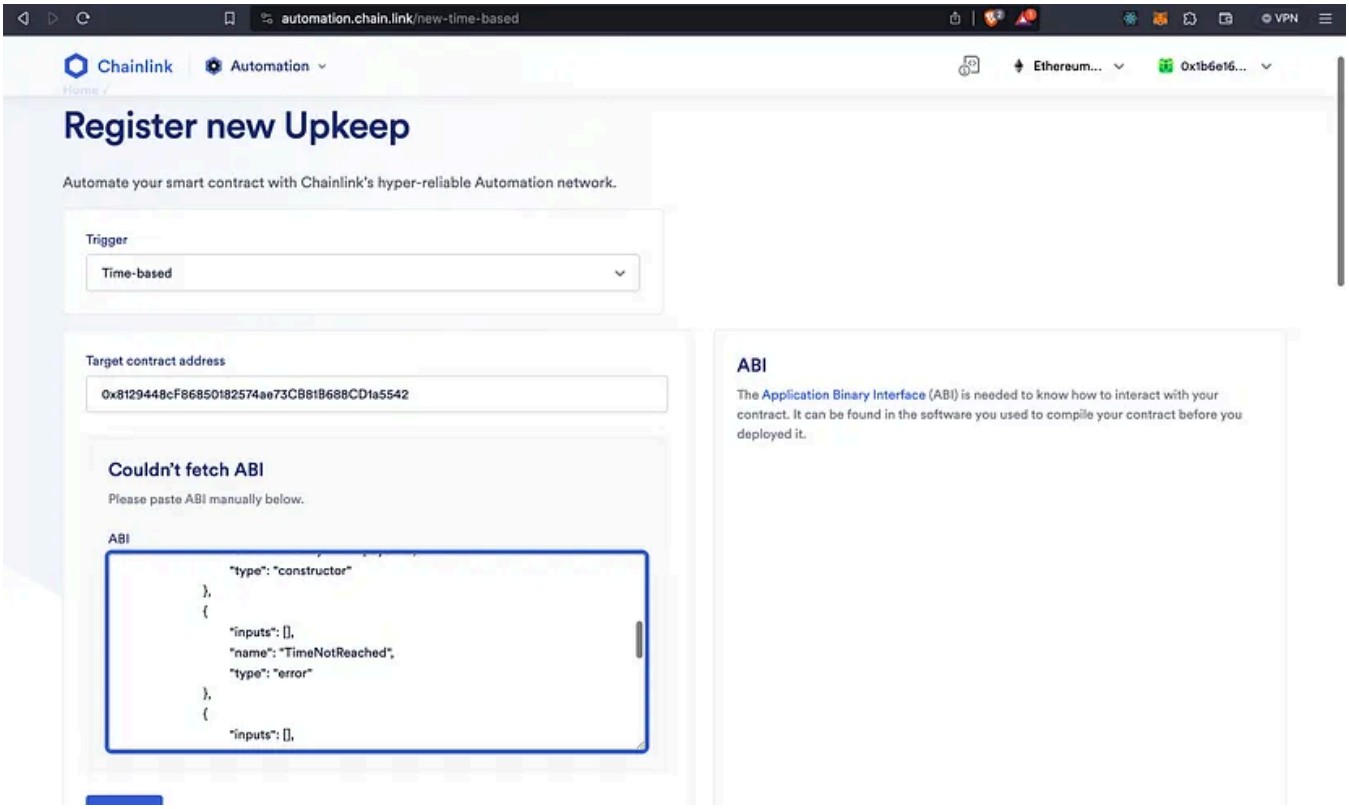
Proceed to Chainlink Automation to register a new upkeep.



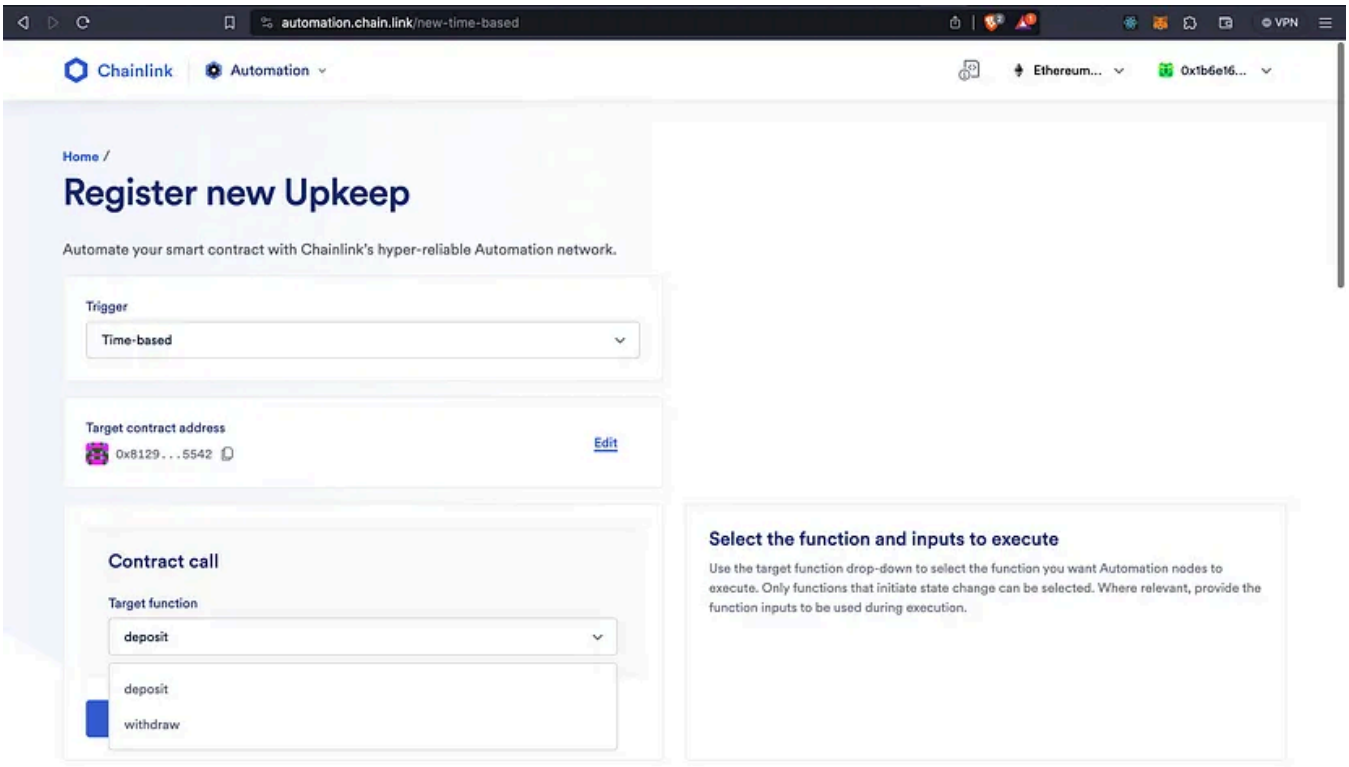
Then select the type of trigger mechanism you want to implement for your contract. For this implementation, am using Time-based automation.



Moving on, you will add the deployed contract address, as well as the ABI of the compiled contract.

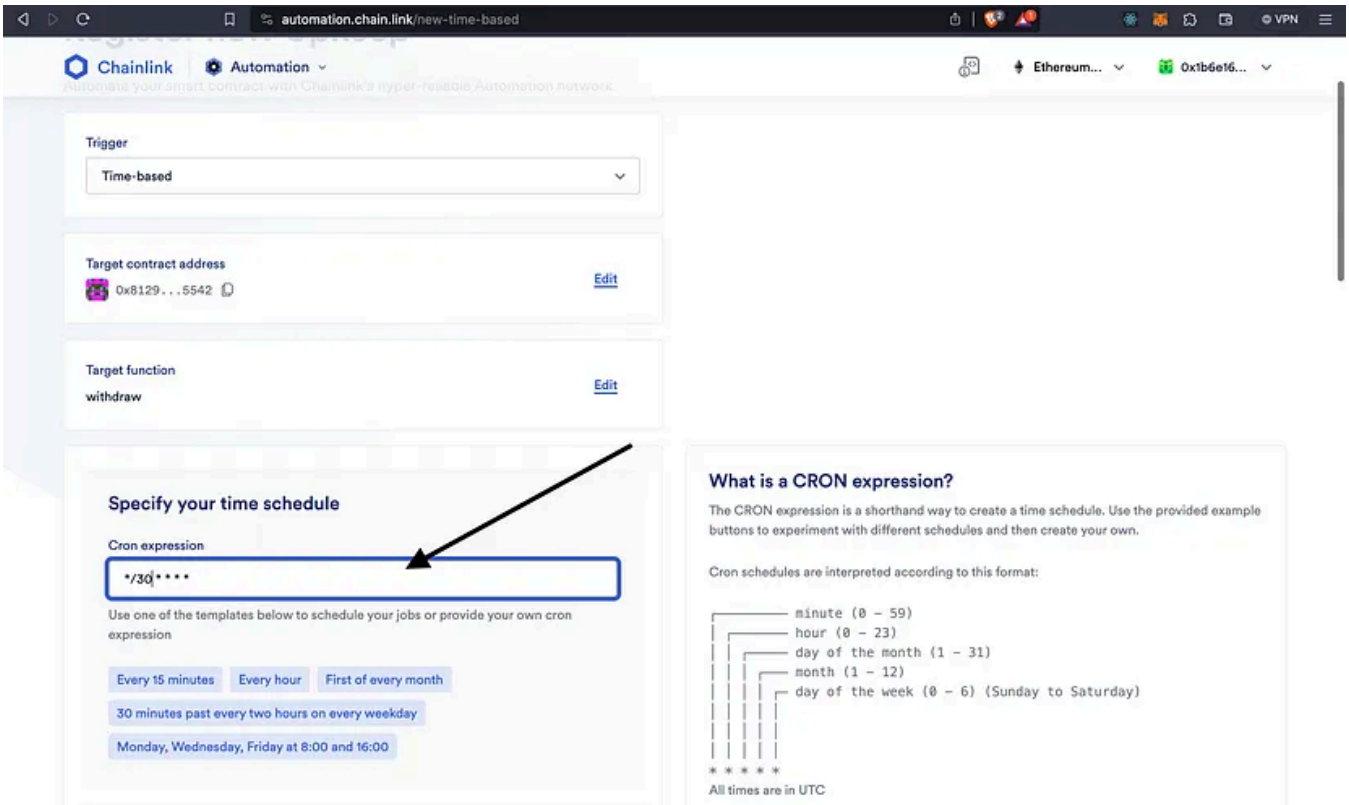


Then you will select the function you want chainlink to automatically trigger when the time you will set reaches.

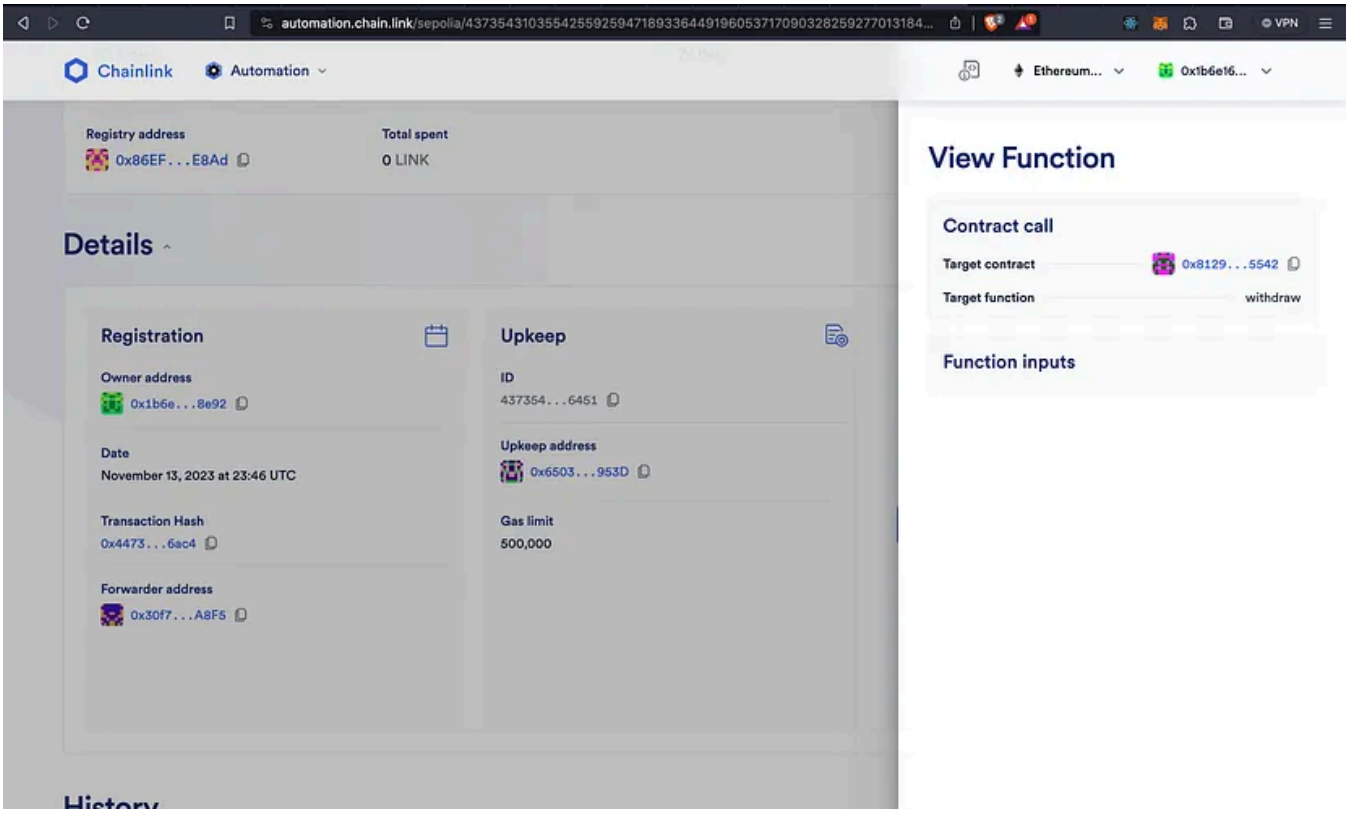


Finally, you will set the time that you want the function to be triggered. Since we are locking the fund till a certain time. You have to add a time that will be after the unlocked time you set when deploying the contract.

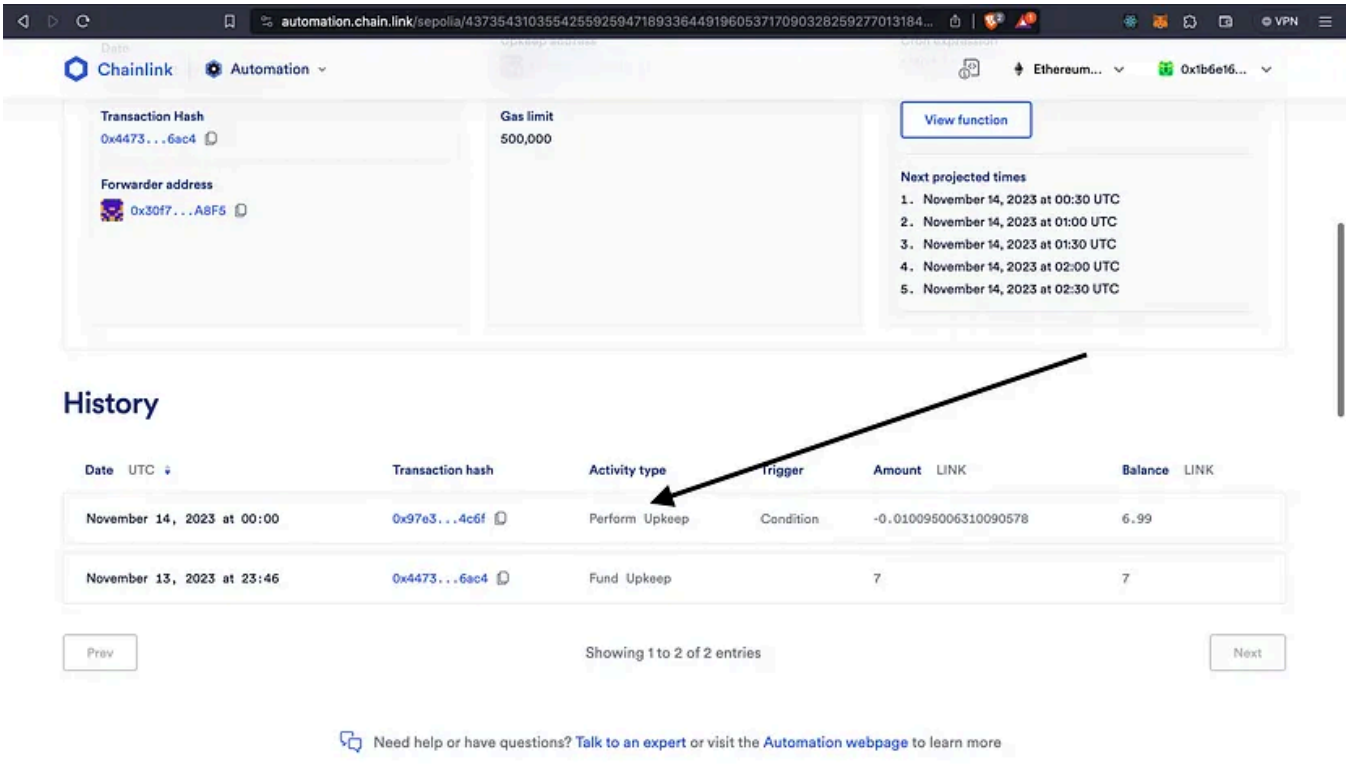
From the image below, you can see that I set a time (30minutes after), then our function will be triggered.



After following every step, you should see an overview like the image below;



After 30 minutes, you will see that the withdraw function was triggered. The contract I wrote and deployed had the withdrawn function triggered by chainlink.



With the illustration above, you can do more with chainlink automation. Different concepts can be implemented using this feature provided by chainlink. You can at anytime, stop the time trigger action, pause and resume the trigger actions, etc.

Applications like staking contracts, lottery games, voting contracts, etc can all be built and have its functions triggered by time, an event or a customized logic.

You can explore other functionalities as well, also try implementing them too.

Chain Link | Smart Contracts Tutorial | Chainlink Automation | Smart Contract Blockchain | Solidity Tutorial

Follow

Written by Joe

82 Followers · Writer for Web3 Writers Guild

Smart Contract. Pisces. Music. Books over People. above all, Gigantura Indica.

More from Joe and Web3 Writers Guild

Joe

Signature Verification in Smart Contracts

I always like to start explaining things from a real world perspective. Let's say you get a signed message from a person.

3 min read · Oct 9, 2023

85

Natachi Nnamaka in Web3 Writers Guild

Learning From The Biggest DeFi Hacks of January 2024

Introduction

9 min read · Feb 6, 2024

64

Esther Oche in Web3 Writers Guild

Step-by-Step process of configuring and deploying your own Arbitrum Orbit chain.

Introduction

7 min read · Dec 8, 2023

117 1

Joe

Using ethers.js to get an ERC20 token details

In this article, I will be showing you how to set up ethers, and how to write a script that shows an ERC20 token symbol, decimal, supply...

4 min read · Aug 15, 2023

6

See all from Joe

See all from Web3 Writers Guild

Recommended from Medium

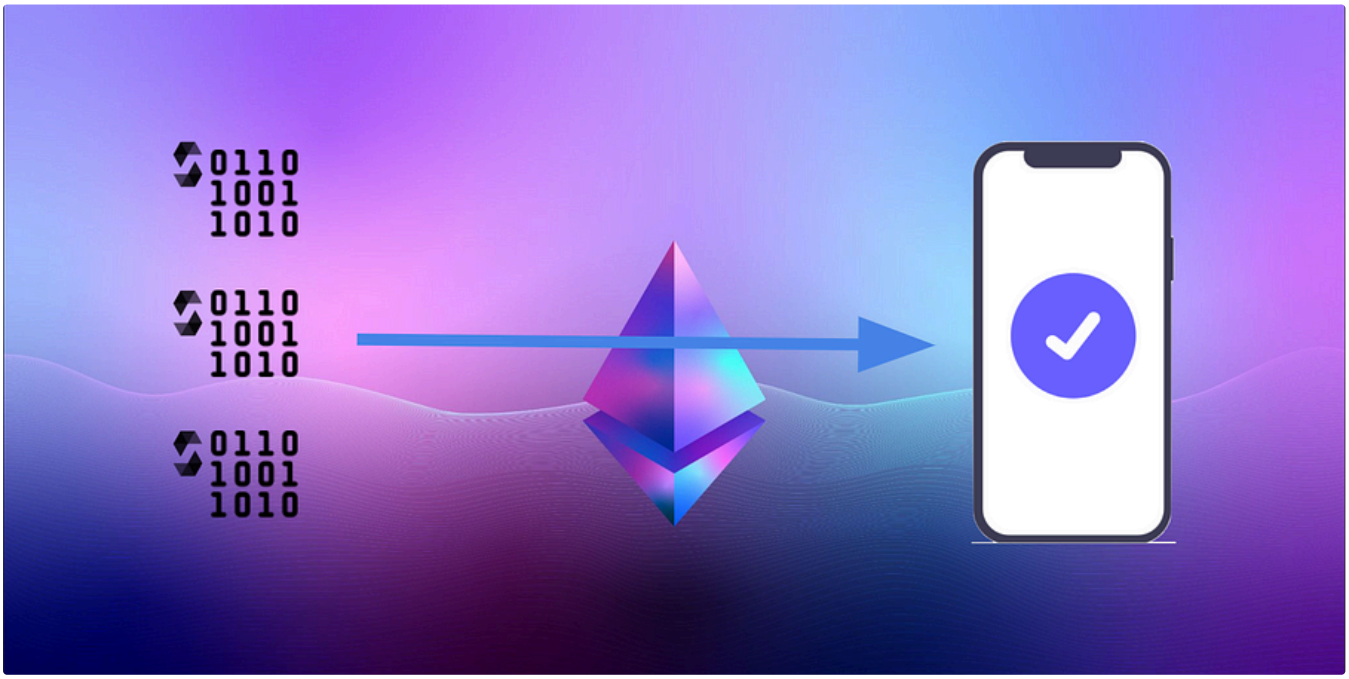
BuildBear Team in BuildBear Labs

Integrating Uniswap V4 into Your Smart Contracts: A Step-by-Step Guide

Uniswap V4 represents a significant leap forward for the Uniswap protocol, enhancing the capabilities of decentralized exchanges (DEXs) and...

4 min read · Mar 1, 2024

8



Leonardo Simone Digiorgio in CoinsBench

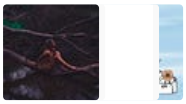
Optimizing Smart Contract Interactions: A Guide to Multi Call integration in Web3 Front-End DApp

Each transaction incurs a gas fee. Executing calls individually is costly. Multi Call reduces expenses, offering an efficient solution.

5 min read · Oct 16, 2023

111 1

Lists



Staff Picks

609 stories · 872 saves

Stories to Help You Level-Up at Work

19 stories · 543 saves

Self-Improvement 101

20 stories · 1540 saves

Productivity 101

20 stories · 1430 saves

GraphLinq

Create Your Own ERC404 Token Without Coding in Five Minutes

In the dynamic world of cryptocurrency, a new trend is shaking things up: the ERC404 token standard.

4 min read · Feb 17, 2024

112

Hanif Olayiwola

A Developer's Intro to Wormhole and Product Ideas

This a detailed deep dive into developing on Wormhole with 5 potential product ideas that can be built on it

41 min read · Mar 17, 2024

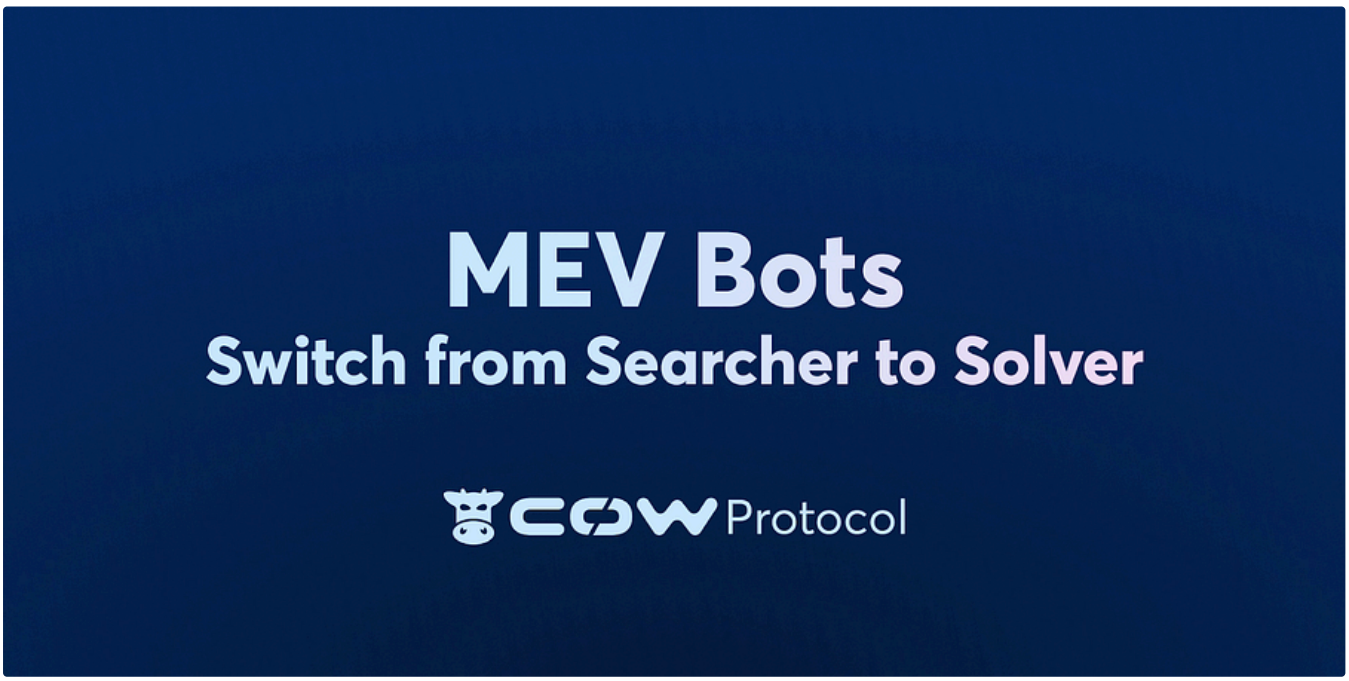
Sena aslıbay

10 Most Common Ethereum Smart Contract Vulnerabilities

1.Missing Access Control

9 min read · Oct 11, 2023

54



CoW Protocol

MEV Bots: Switch From Searcher to Solver

Searchers make Ethereum worse for everyone, but with just a few tweaks they could contribute to the ecosystem instead

4 min read · Mar 8, 2024

66

See more recommendations