OpenZeppelin Chainlink

Smart Contracts Automation

Managing **Upkeeps** on the **Chainlink Keeper** network via **OpenZeppelin Defender**

Martín Verzilli
mverzilli@openzeppelin.com
🐦 @mverzilli

Patrick Collins
patrick.collins@chainlinklabs.com
🐦 @PatrickAlphaC

OpenZeppelin

Our mission is to protect the open economy

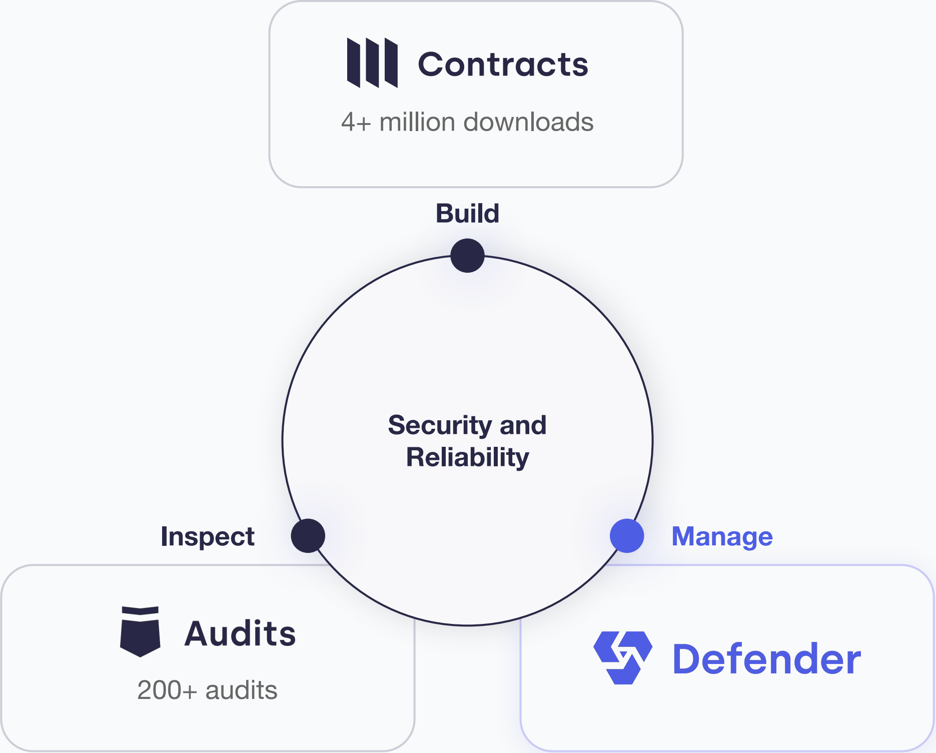
OpenZeppelin is a software company that provides **security audits** and **products** for decentralized systems.

Projects from any size -from new startups to established organizations- trust OpenZeppelin to build, inspect and connect to the open economy.

Compound
coinbase
AAVE
δY/δX
brave
UMA
Balancer
ethereum foundation
augur
celo
CIRCLE
BitGo
pool together
Set
opyn

Security, Reliability and Risk Management

OpenZeppelin provides a complete suite of **security and reliability products** to build, manage, and inspect all aspects of software development and operations for Ethereum projects.



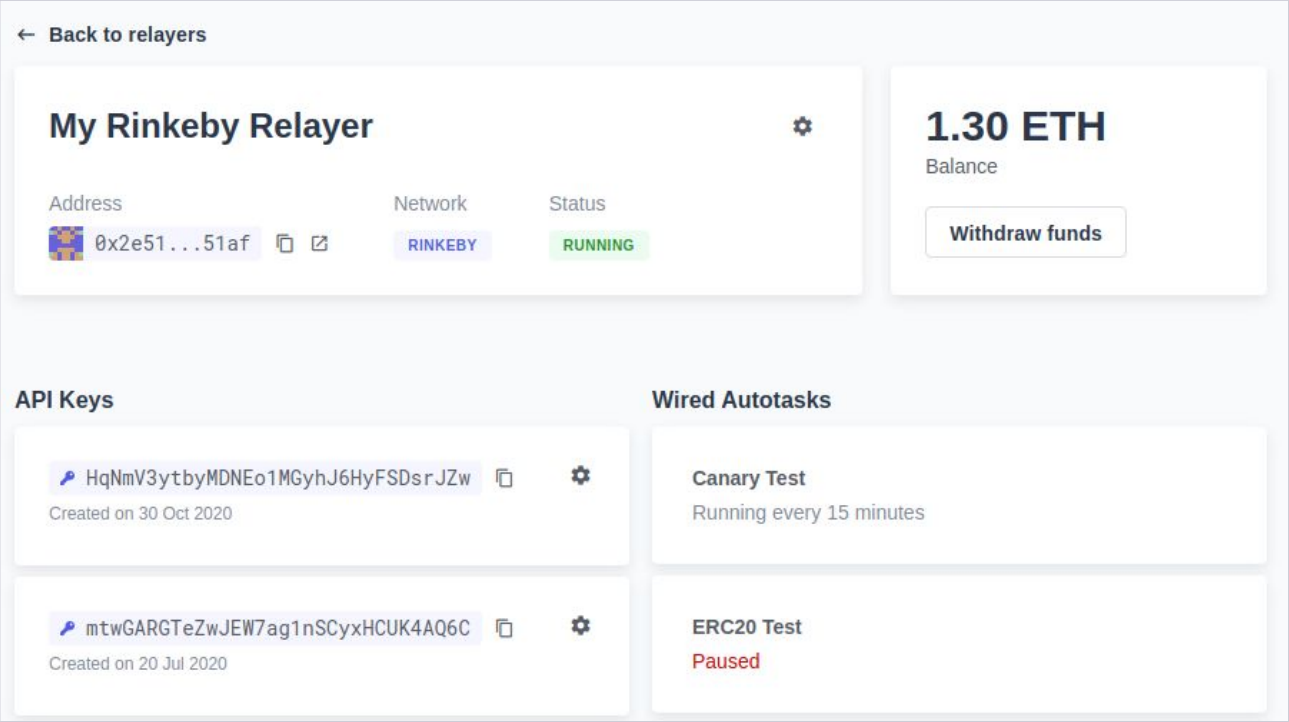
Automating smart contract operations

Sending transactions, monitoring, administration, funding, etc

Scheduling operations / Crontab

```
$ crontab -e
# m h dom mon dow command
0 5 * * * /usr/bin/node ~/scripts/update-contract.js

$ cat ~/scripts/.env
PK=0x74b0944e1f379af54f36312652f4c19b34c...
```

Scheduling operations / Using Defender



← Back to relayers

My Rinkeby Relayer

Address: 0x2e51...51af

Network: RINKEBY

Status: RUNNING

1.30 ETH

Balance

Withdraw funds

API Keys

HqNmV3ytbyMDNEo1MGyhJ6HyFSDsrJZw
Created on 30 Oct 2020

mtwGARGTeZwJEW7ag1nSCyxHCUK4AQ6C
Created on 20 Jul 2020

Wired Autotasks

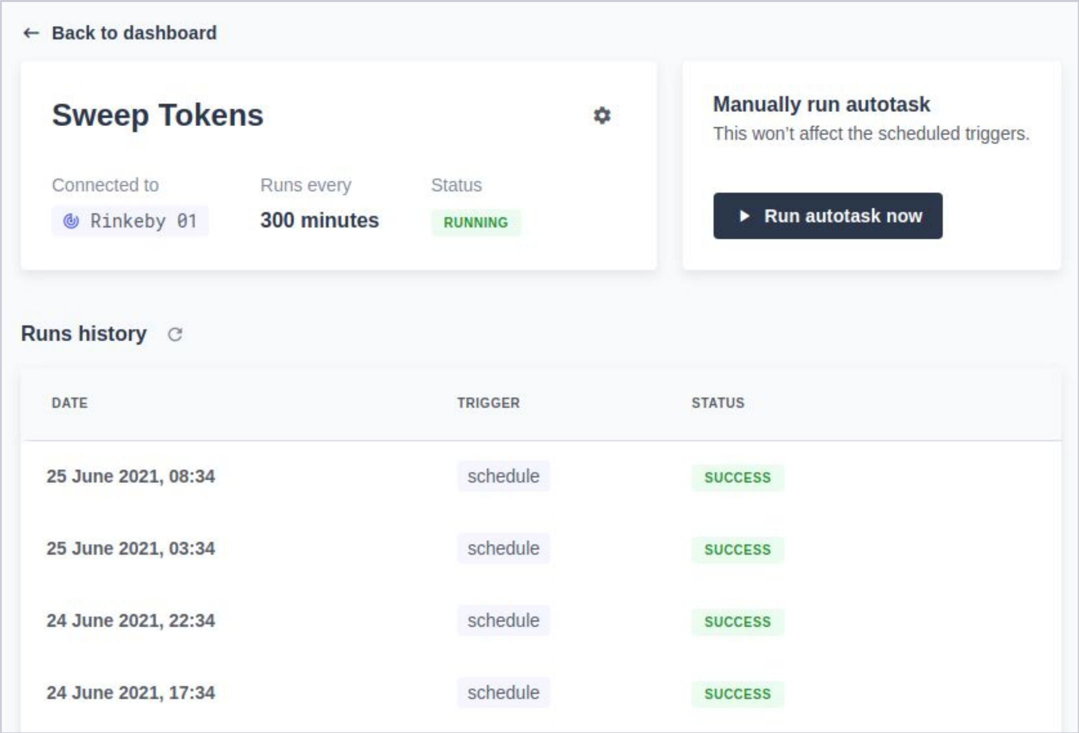
Canary Test
Running every 15 minutes

ERC20 Test
Paused

Secure vault for private keys with Defender Relayers

OpenZeppelin

Scheduling operations / Using Defender



← Back to dashboard

Sweep Tokens

Connected to: Rinkeby 01

Runs every: 300 minutes

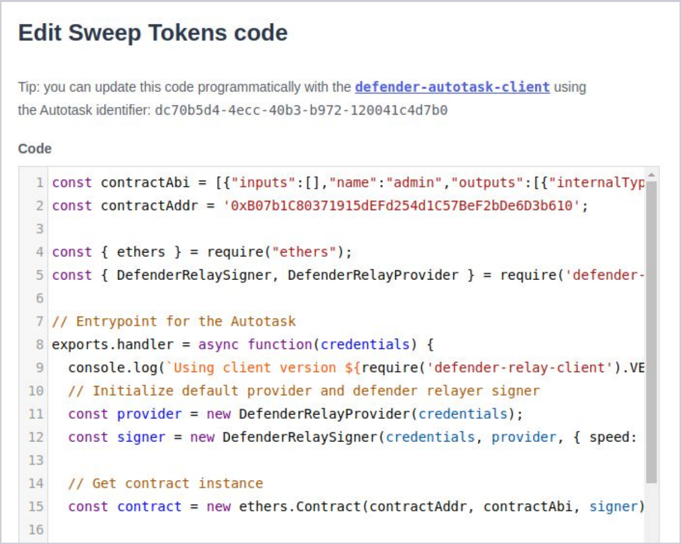
Status: RUNNING

Manually run autotask

This won't affect the scheduled triggers.

Runs history

DATE	TRIGGER	STATUS
25 June 2021, 08:34	schedule	SUCCESS
25 June 2021, 03:34	schedule	SUCCESS
24 June 2021, 22:34	schedule	SUCCESS
24 June 2021, 17:34	schedule	SUCCESS



Edit Sweep Tokens code

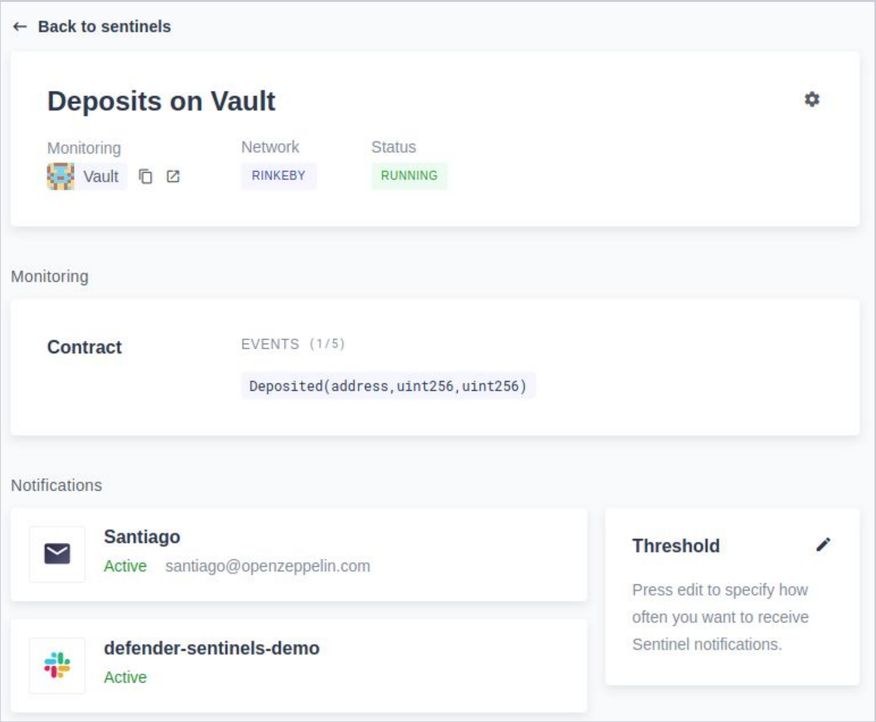
Tip: you can update this code programmatically with the `defender-autotask-client` using the Autotask identifier: `dc70b5d4-4ecc-40b3-b972-120041c4d7b0`

Code

```
const contractAbi = [{"inputs":[],"name":"admin","outputs":[{"internalTyp
const contractAddr = '0xB07b1C80371915dEFd254d1C57BeF2bDe6D3b610';

const { ethers } = require("ethers");
const { DefenderRelaySigner, DefenderRelayProvider } = require('defender-

// Entrypoint for the Autotask
exports.handler = async function(credentials) {
  console.log(`Using client version ${require('defender-relay-client').VE
  // Initialize default provider and defender relayer signer
  const provider = new DefenderRelayProvider(credentials);
  const signer = new DefenderRelaySigner(credentials, provider, { speed:

  // Get contract instance
  const contract = new ethers.Contract(contractAddr, contractAbi, signer)

```

Scheduling scripts execution with Defender Autotasks

Scheduling operations / Using Defender



← Back to sentinels

Deposits on Vault

Monitoring: Vault

Network: RINKEBY

Status: RUNNING

Monitoring

Contract — EVENTS (1/5)

`Deposited(address,uint256,uint256)`

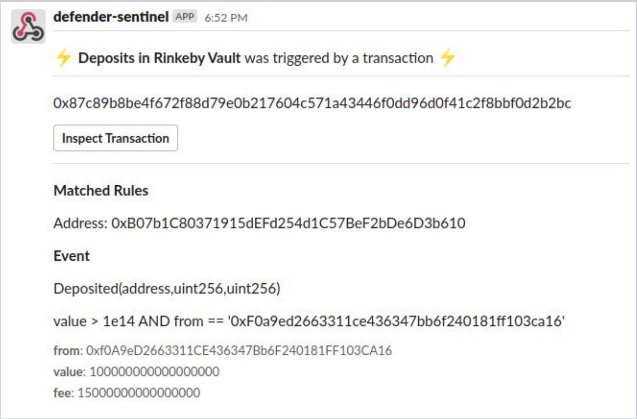
Notifications

Santiago
Active santiago@openzeppelin.com

defender-sentinels-demo
Active

Threshold

Press edit to specify how often you want to receive Sentinel notifications.



defender-sentinel APP 6:52 PM

⚡ **Deposits in Rinkeby Vault** was triggered by a transaction ⚡

0x87c89b8be4f672f88d79e0b217604c571a43446f0dd96d0f41c2f8bbf0d2b2bc

Inspect Transaction

Matched Rules

Address: 0xB07b1C80371915dEFd254d1C57BeF2bDe6D3b610

Event

Deposited(address,uint256,uint256)

value > 1e14 AND from == '0xF0a9ed2663311ce436347bb6f240181ff103ca16'

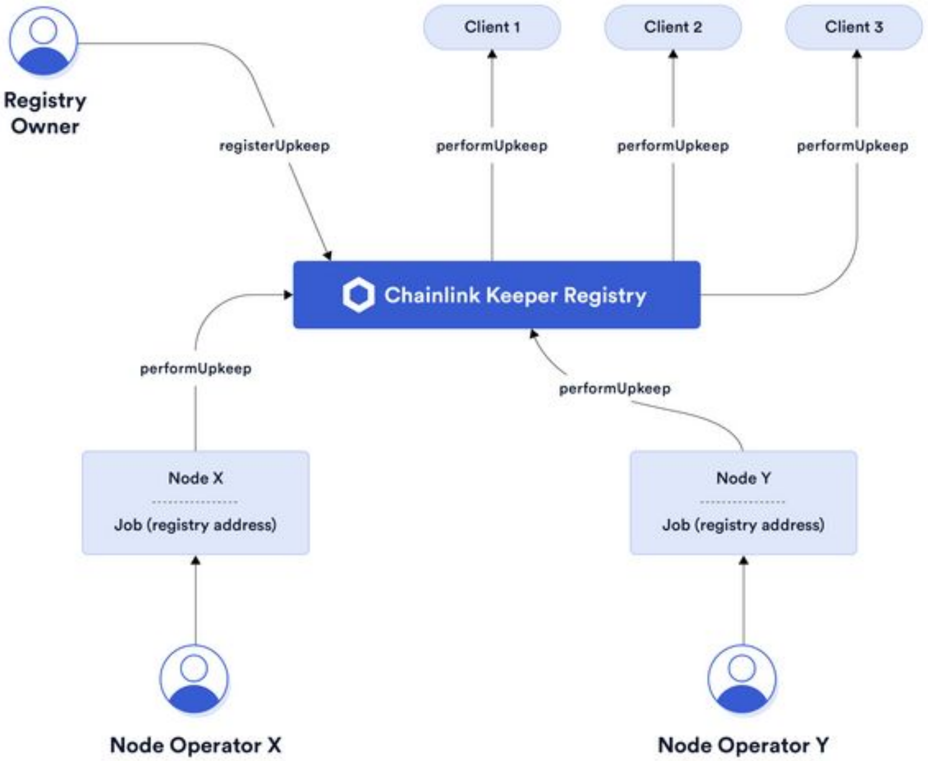
from: 0xf0A9eD2663311CE436347Bb6F240181FF103CA16
value: 1000000000000000000
fee: 150000000000000000

Monitoring contracts with Defender Sentinels

Distributing the workload

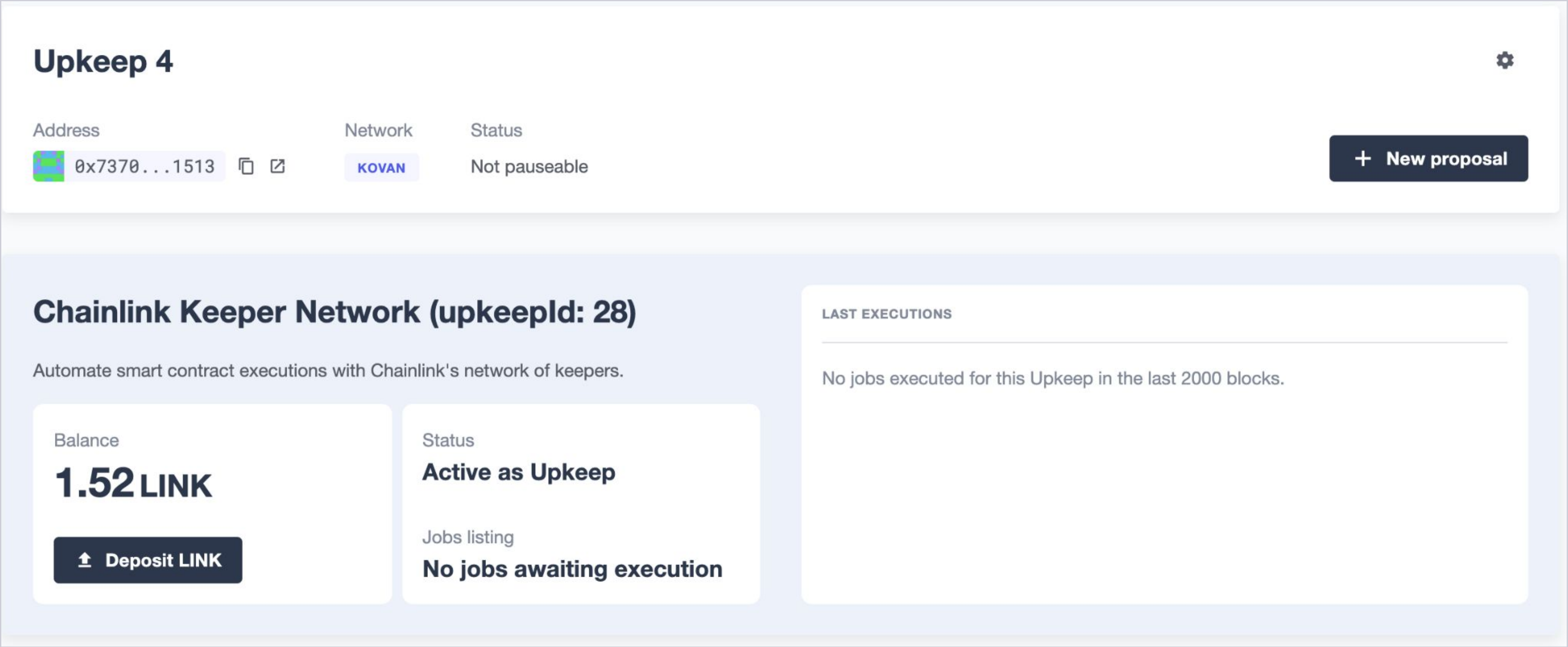
Use a distributed **network** of **incentivized** workers to execute all smart contract operations

Scheduling operations / Using Chainlink Keepers via Defender



OpenZeppelin

Scheduling operations / Using Chainlink Keepers via Defender



Upkeep 4

Address: 0x7370...1513

Network: KOVAN

Status: Not pauseable

Chainlink Keeper Network (upkeepId: 28)

Automate smart contract executions with Chainlink's network of keepers.

Balance: 1.52 LINK

Status: Active as Upkeep

Jobs listing: No jobs awaiting execution

LAST EXECUTIONS

No jobs executed for this Upkeep in the last 2000 blocks.

OpenZeppelin

Patrick Collins
Developer Advocate

Chainlink

Martin Verzilli
Software Engineer

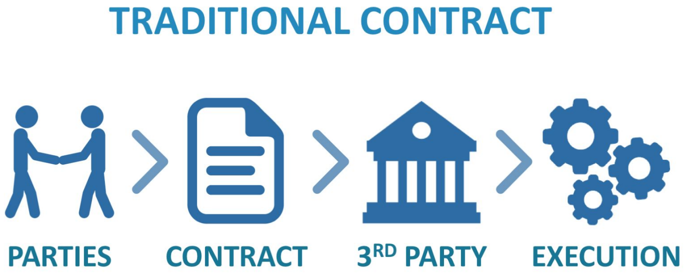
OpenZeppelin

What is the Chainlink Keeper Network

Patrick Collins

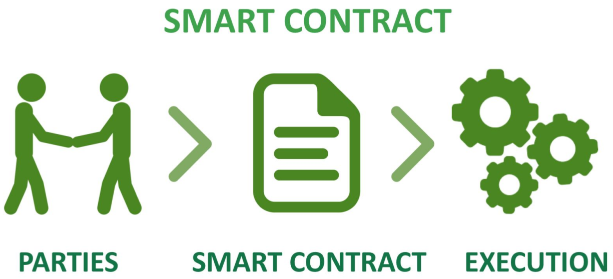
Chainlink Keepers Network

What are Smart Contracts



TRADITIONAL CONTRACT

PARTIES > CONTRACT > 3RD PARTY > EXECUTION



SMART CONTRACT

PARTIES > SMART CONTRACT > EXECUTION

Problems with Smart Contracts

Off Chain Data : *disconnected from external resources*

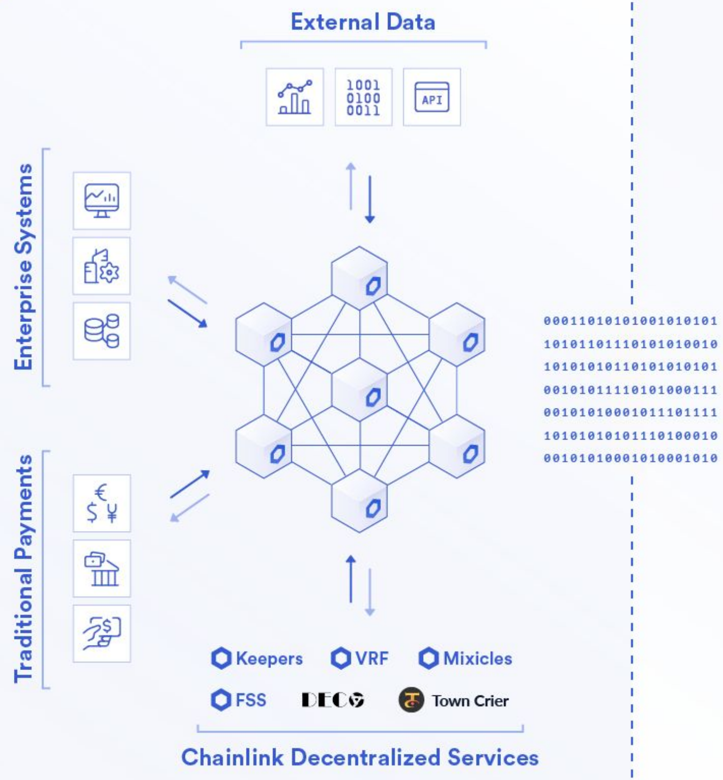
- Enterprise data
- API data
- Real world data (weather, current affairs, other facts)

Off chain computation

- Scalable computation : *costly*
- Regular maintenance tasks : *asleep by default*
- Computation services (Generate Random numbers, proof of reserve etc)

Off-chain Data and Computation

On-chain Data and Computation



External Data

Enterprise Systems

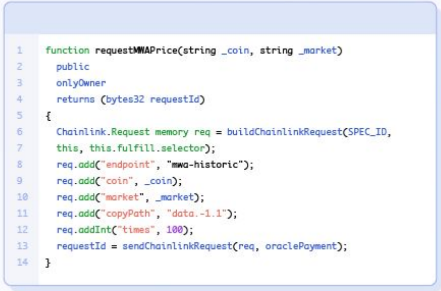
Traditional Payments

Keepers VRF Mixicles

FSS DECO Town Crier

Chainlink Decentralized Services

Smart Contract On-Chain Code



```
function requestMWAPrice(string _coin, string _market)
  public
  onlyOwner
  returns (bytes32 requestId)
{
  Chainlink.Request memory req = buildChainlinkRequest(SPEC_ID,
  this, this.fulfill.selector);
  req.add("endpoint", "mwa-historic");
  req.add("coin", _coin);
  req.add("market", _market);
  req.add("copyPath", "data.-1.1");
  req.addInt("times", 100);
  requestId = sendChainlinkRequest(req, oraclePayment);
}
```

bitcoin ethereum HYPERLEDGER

Tezos Polkadot. Kava

BINANCE SMART CHAIN CONFLUX COSMOS

polygon

Smart Contract Platforms

Why Keepers?

- **External triggers are necessary in various use cases**
- On-chain conditions must be met to trigger functions
- So what are your options?
  - Build software & infrastructure yourself
  - Support it 24/7
  - Make sure it doesn’t miss a beat
- … So why not pay someone else to do it?

What it is like today?

- Centralized 3rd party solution
- Open systems
  - take the risk of bots aggressively competing with one another on gas price
- Your upkeep may not be completed in some models given too low a bounty
- Economic incentives are not always well aligned for stability/consistency

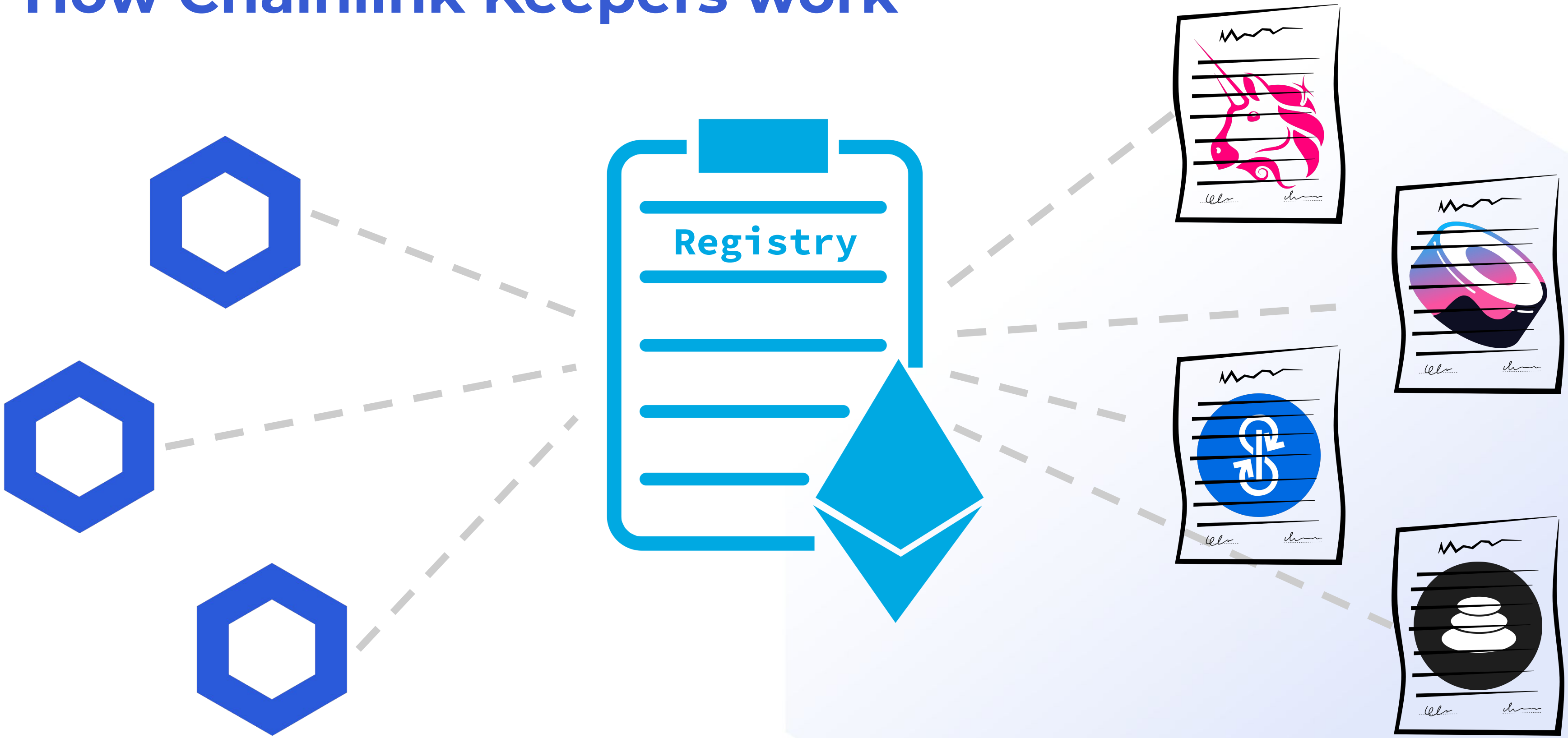
Why Chainlink Keepers

- High Uptime *: professional devops*
- Low Costs *: payments model*
- Decentralized Execution *: turn taking algo, all keepers take turns*
- Transparent Reputation

Use Cases

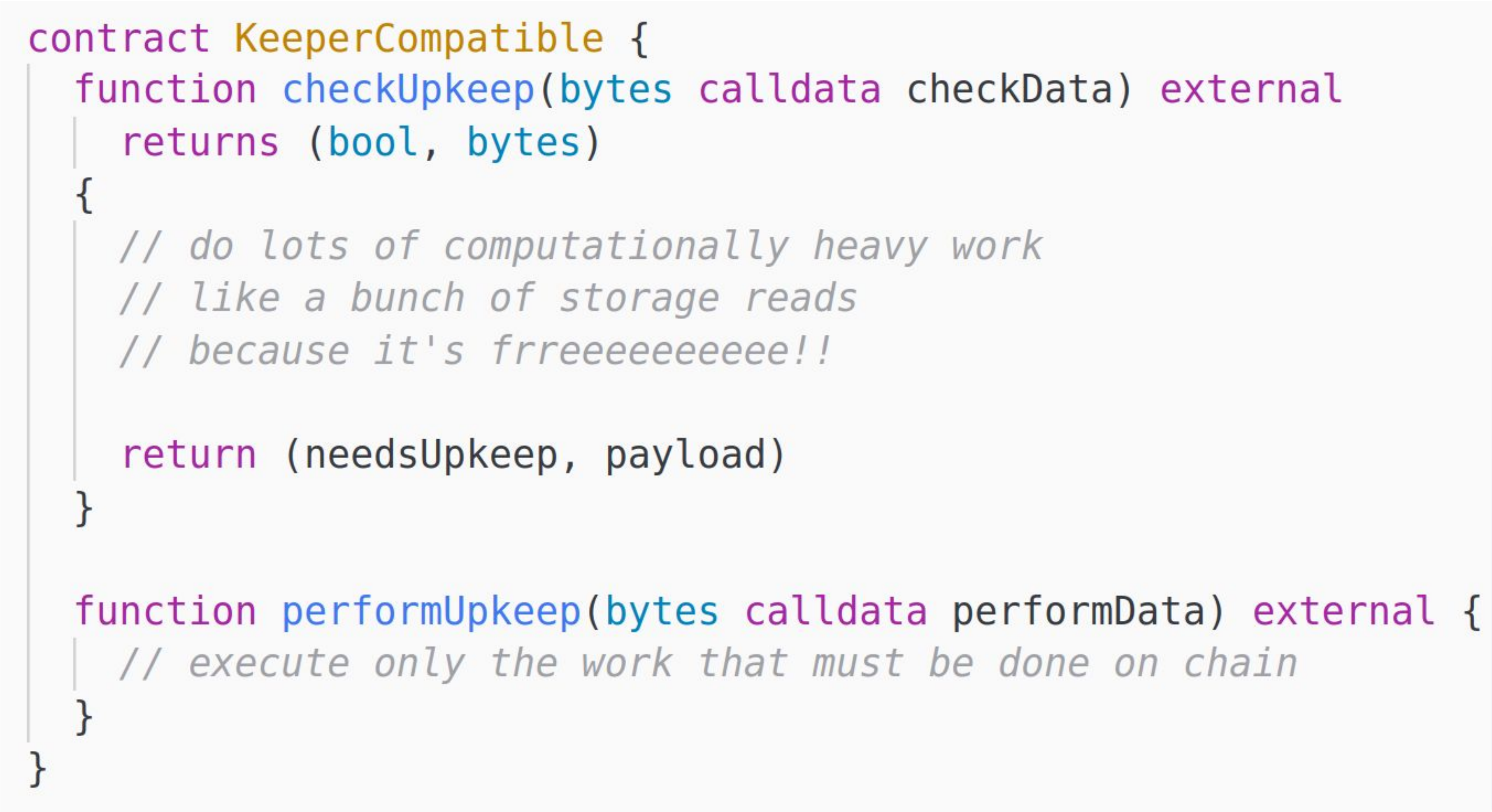
- Execute limit orders on decentralized exchanges
- Mint tokens when reserves increase
- Harvest yield from vaults
- Rebase elastic supply tokens
- Trigger automated trading strategies
- Liquidate undercollateralized loans
- Release locked assets after periods of inactivity
- Top up token balances falling below a minimum threshold

How Chainlink Keepers work



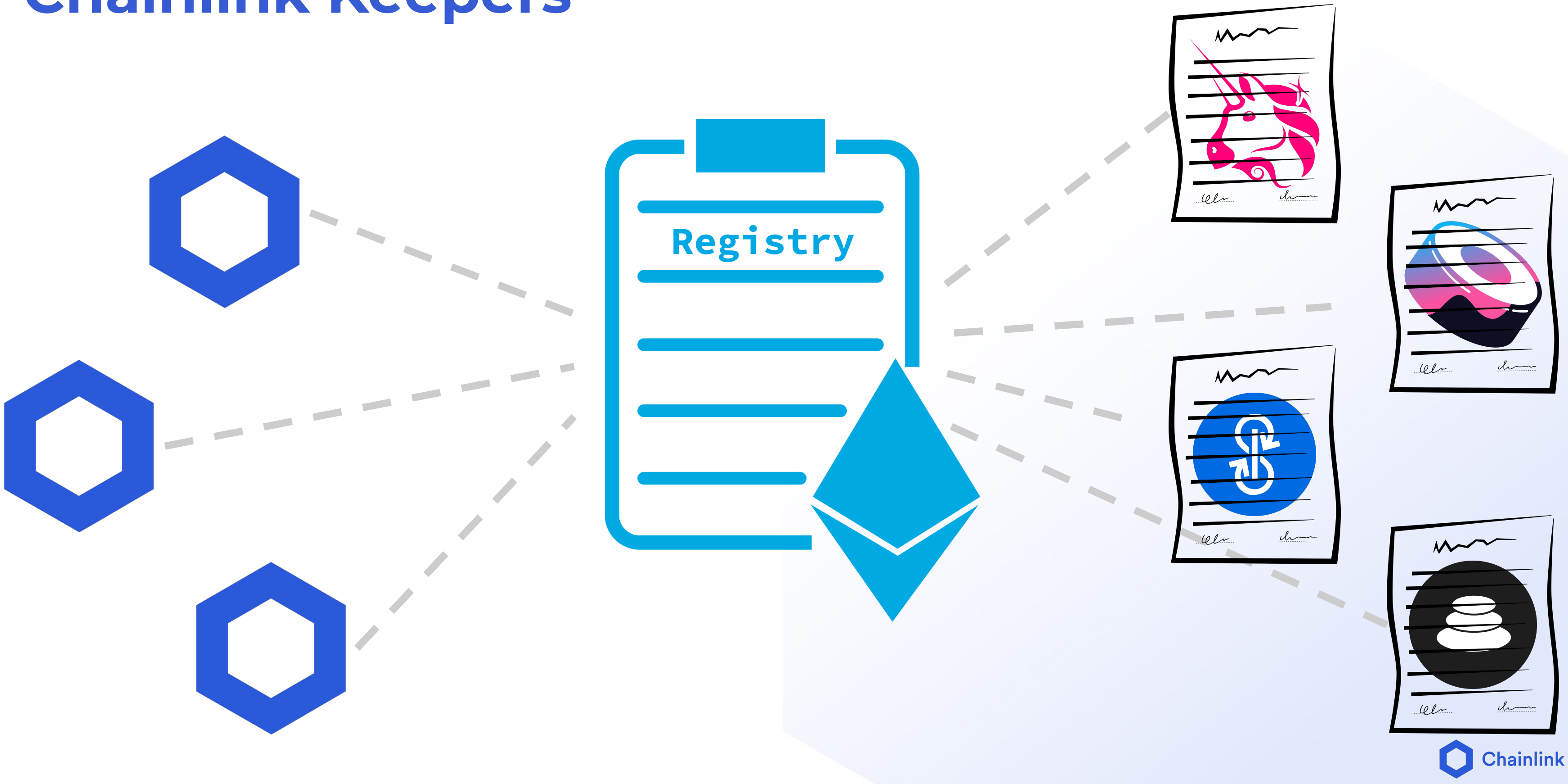
Chainlink Keepers talk to a central Keeper Registry smart contract. Customers register their contracts on the registry.

Chainlink

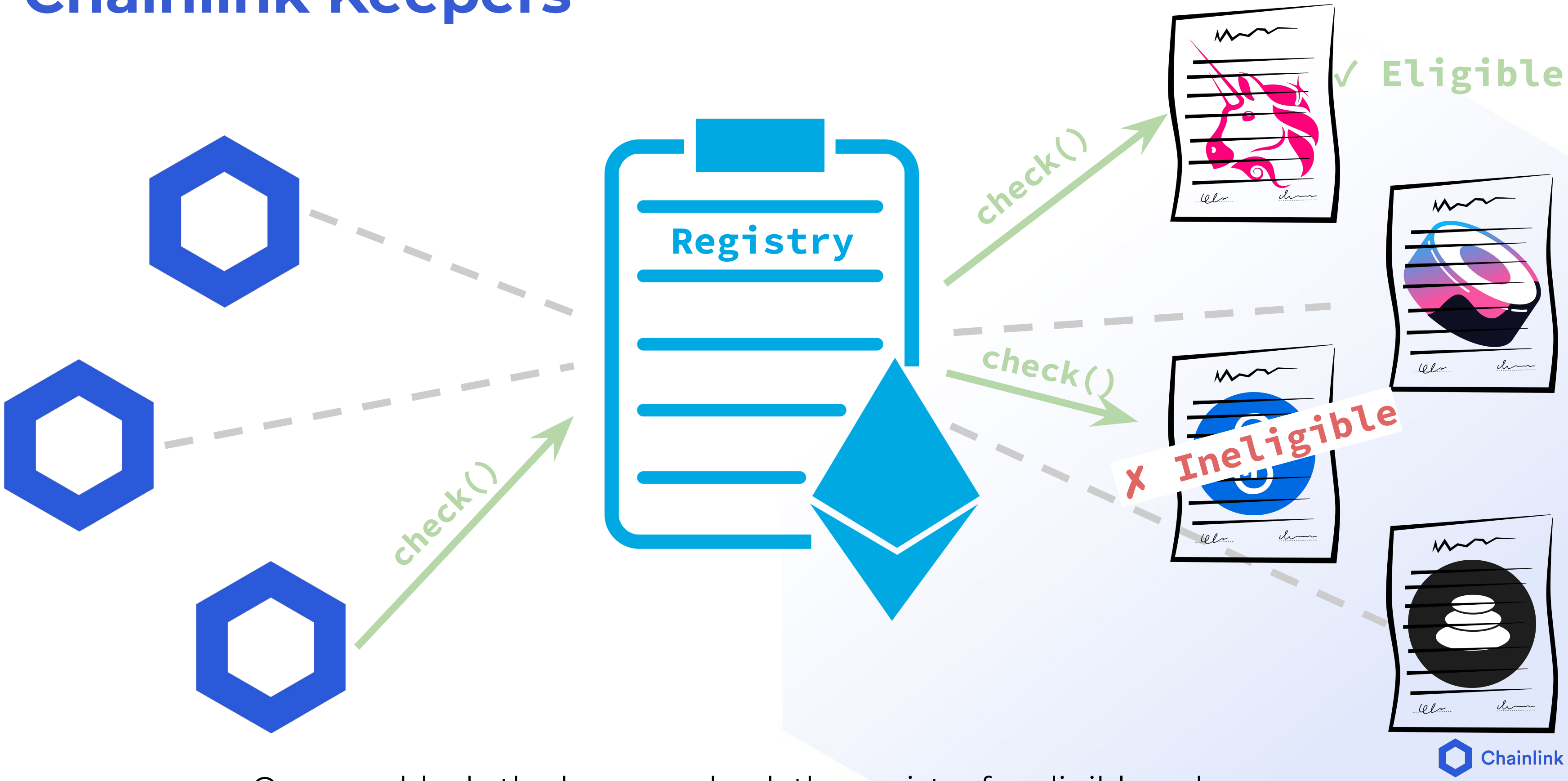


```
contract KeeperCompatible {
  function checkUpkeep(bytes calldata checkData) external
    returns (bool, bytes)
  {
    // do lots of computationally heavy work
    // like a bunch of storage reads
    // because it's frreeeeeeeeee!!

    return (needsUpkeep, payload)
  }

  function performUpkeep(bytes calldata performData) external {
    // execute only the work that must be done on chain
  }
}
```

Chainlink Keepers

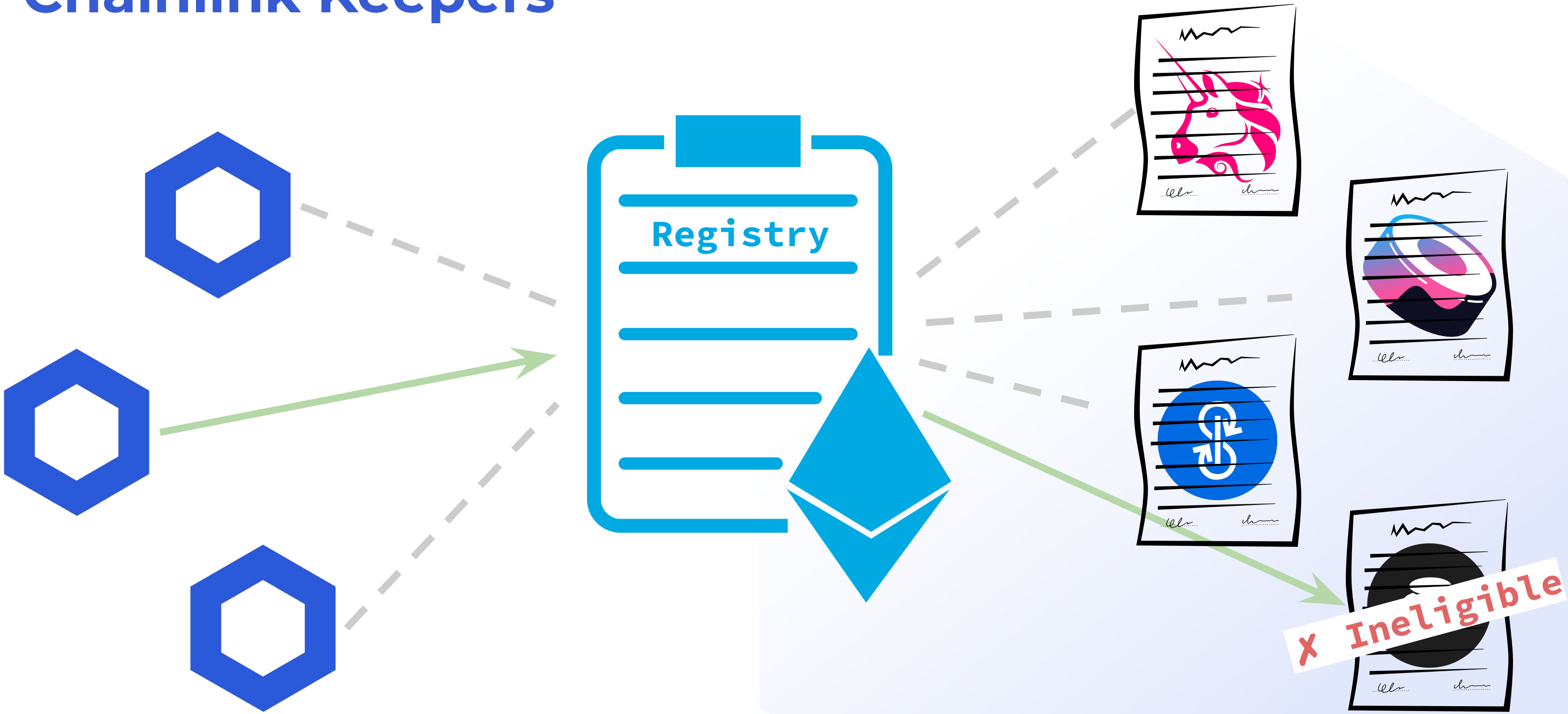


Chainlink Keepers



On every block, the keepers check the registry for eligible upkeeps

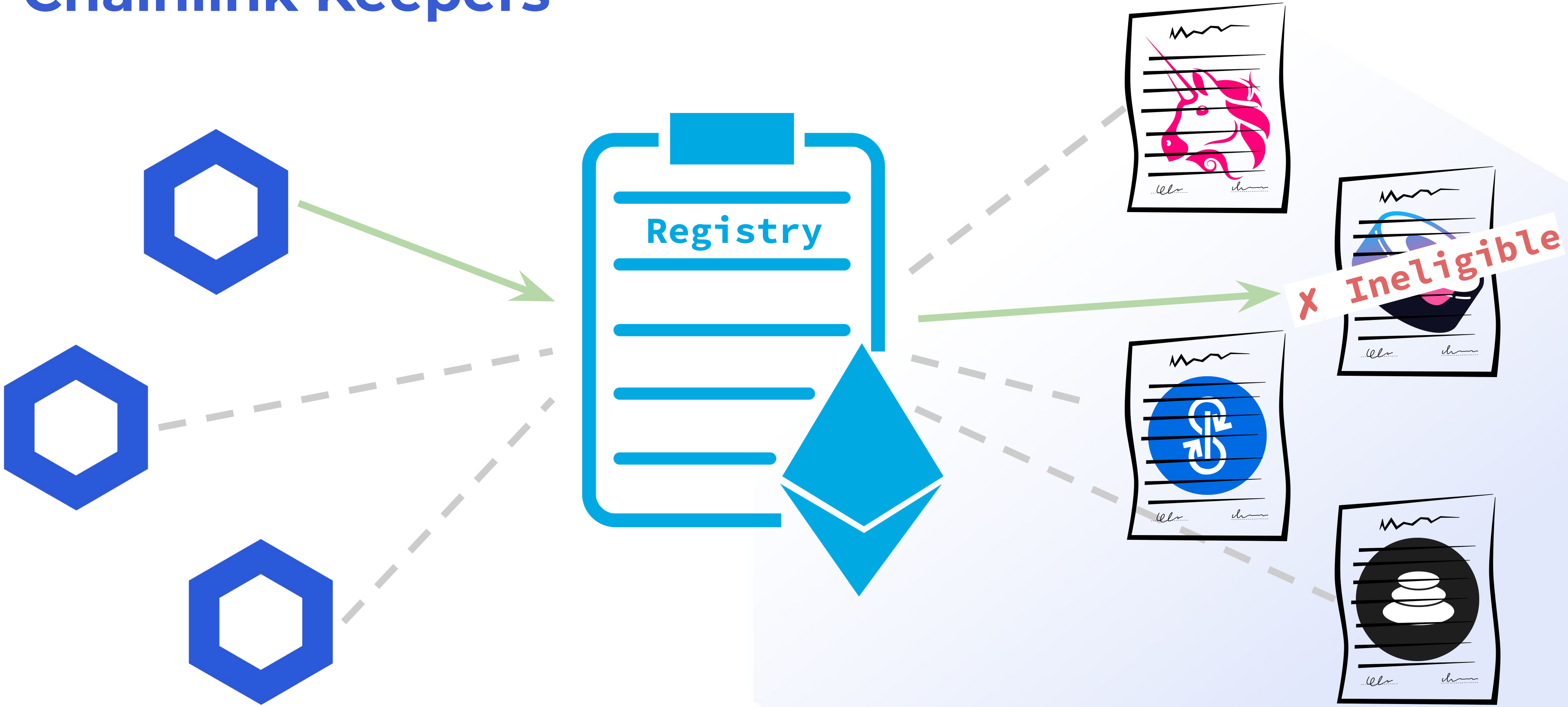
Chainlink Keepers



Different Keepers are assigned different upkeeps
according to a turn taking algorithm

Chainlink

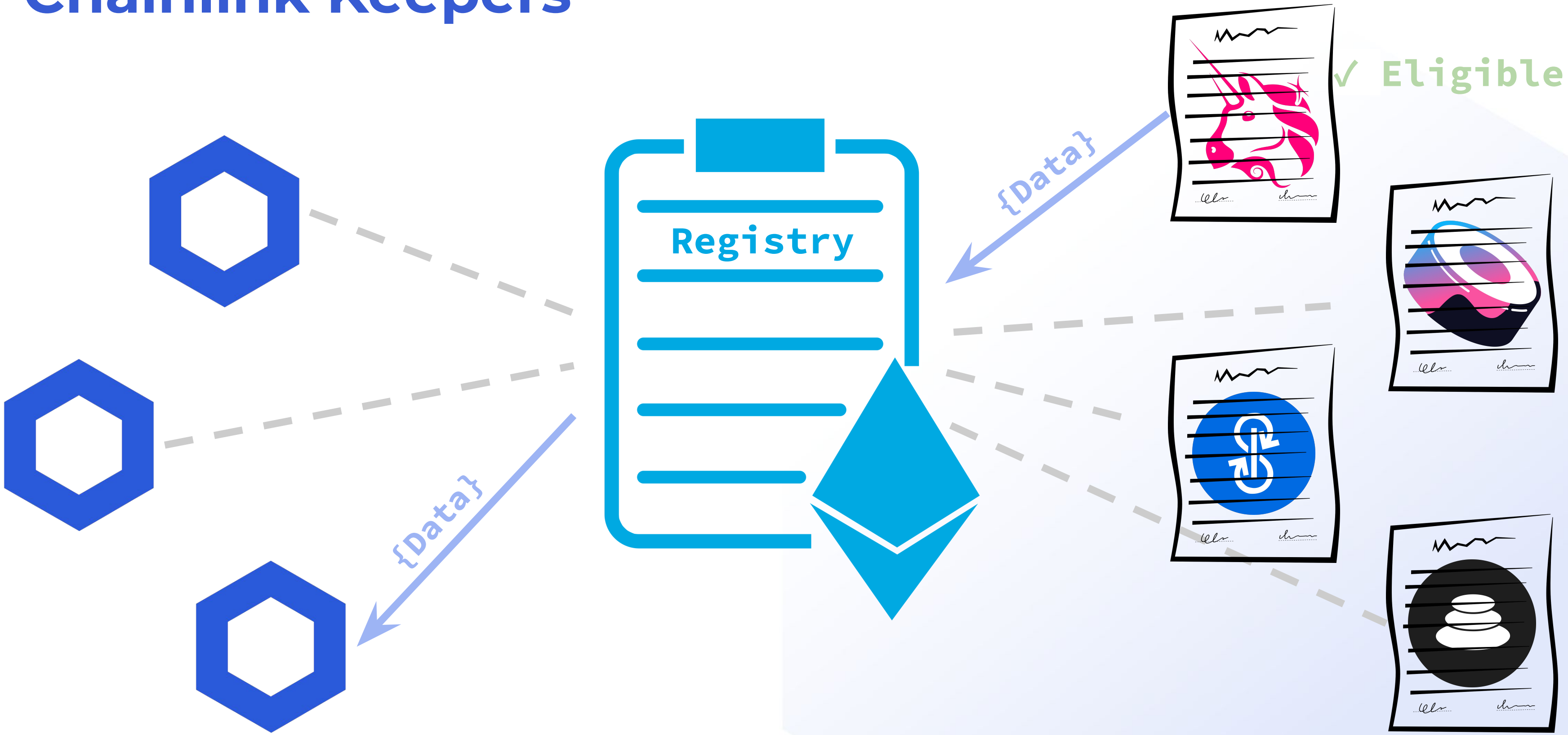
Chainlink Keepers



Different Keepers are assigned different upkeeps
according to a turn taking algorithm

Chainlink

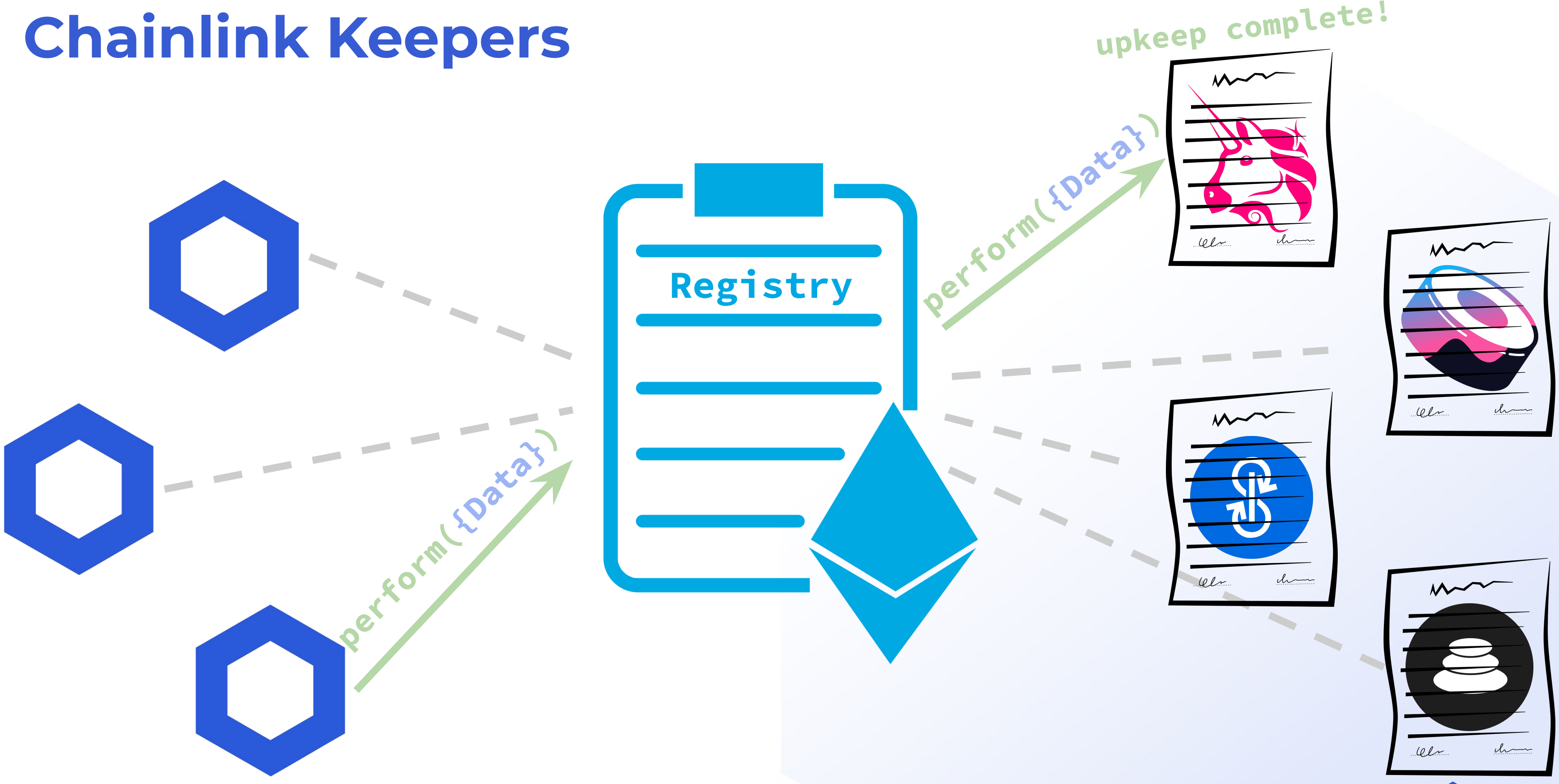
Chainlink Keepers



Chainlink

Eligible upkeeps emit a payload to pass into the perform transaction

Chainlink Keepers



Chainlink

The keeper submits a perform transaction with the data

Demo

- Chainlink Keepers
- OpenZeppelin Defender

Managing Upkeeps from Defender

Martín Verzilli

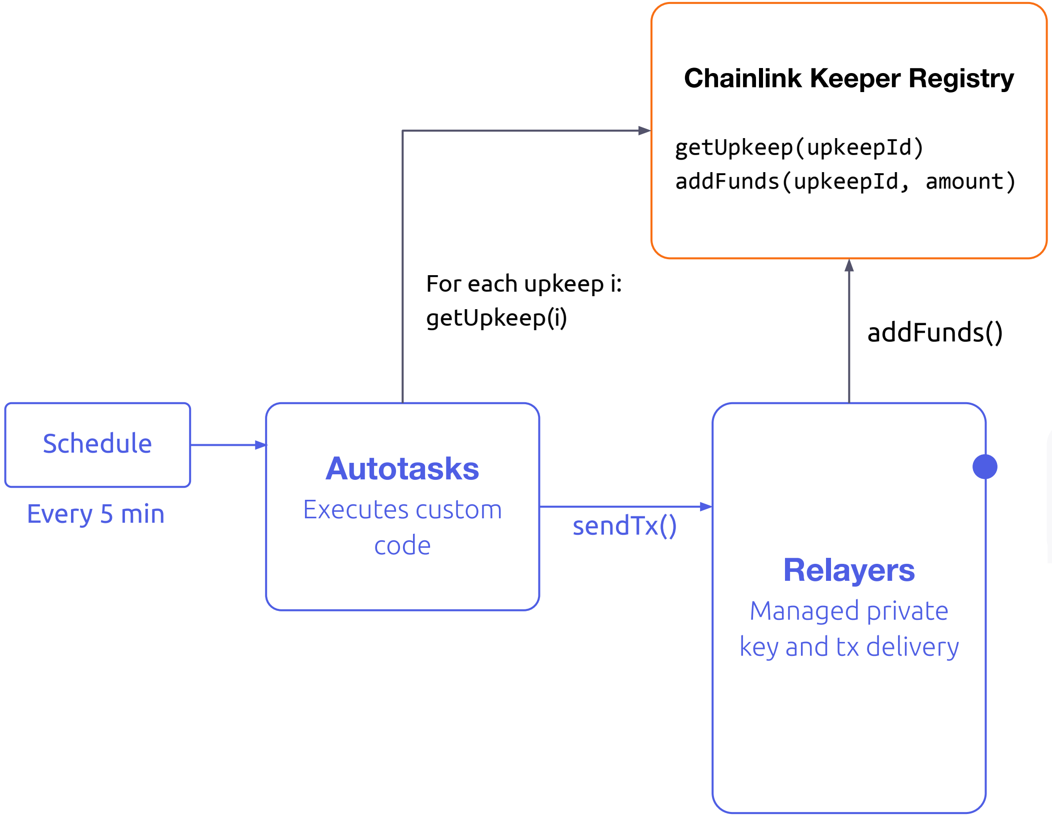
Agenda

1. Register a new Upkeep using Admin.
2. Auto top up your upkeeps with Autotask and Relay.
3. Monitor post-execution invariants combining Sentinels, Autotasks and Relay.

1. Registering an Upkeep

1. Deploy an Upkeep compatible contract.
2. Verify your contract code on Etherscan.
3. Import your contract to Defender.
4. Fill the registration forms.
5. Wait for registration approval.

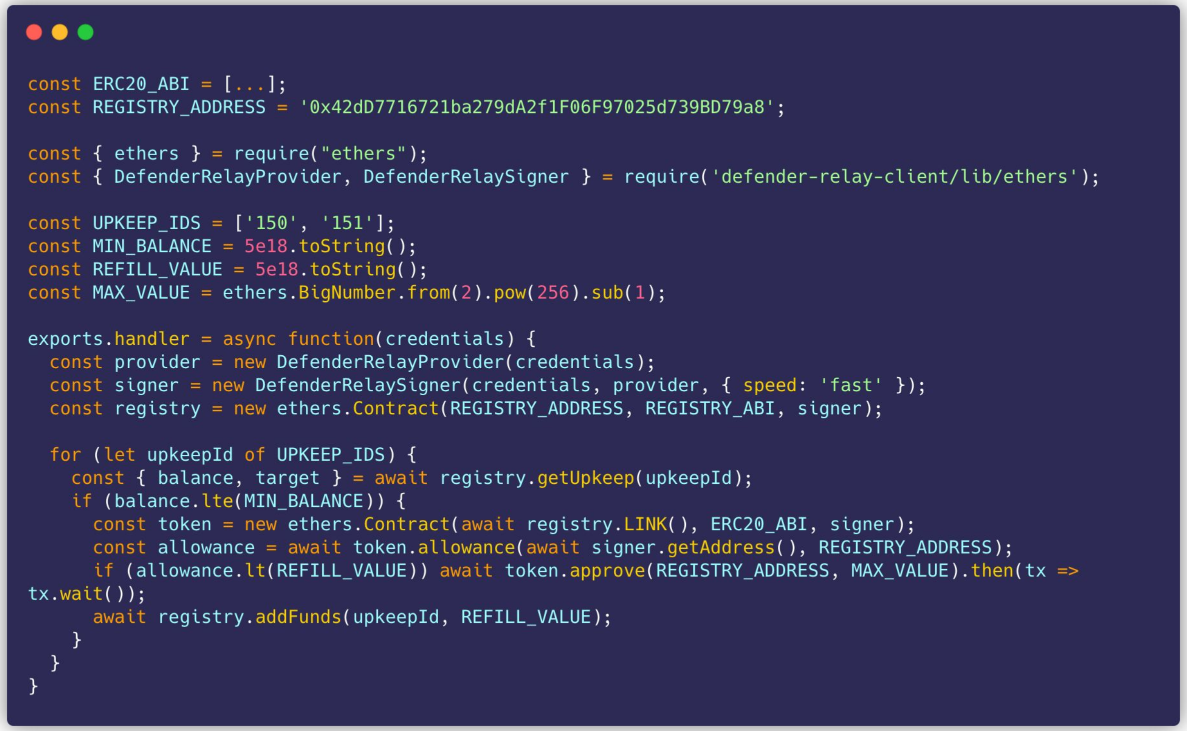
2. Auto funding your Upkeeps



Keep funds in a single relayer, use it to fund multiple upkeeps when balance goes below a given threshold

OpenZeppelin

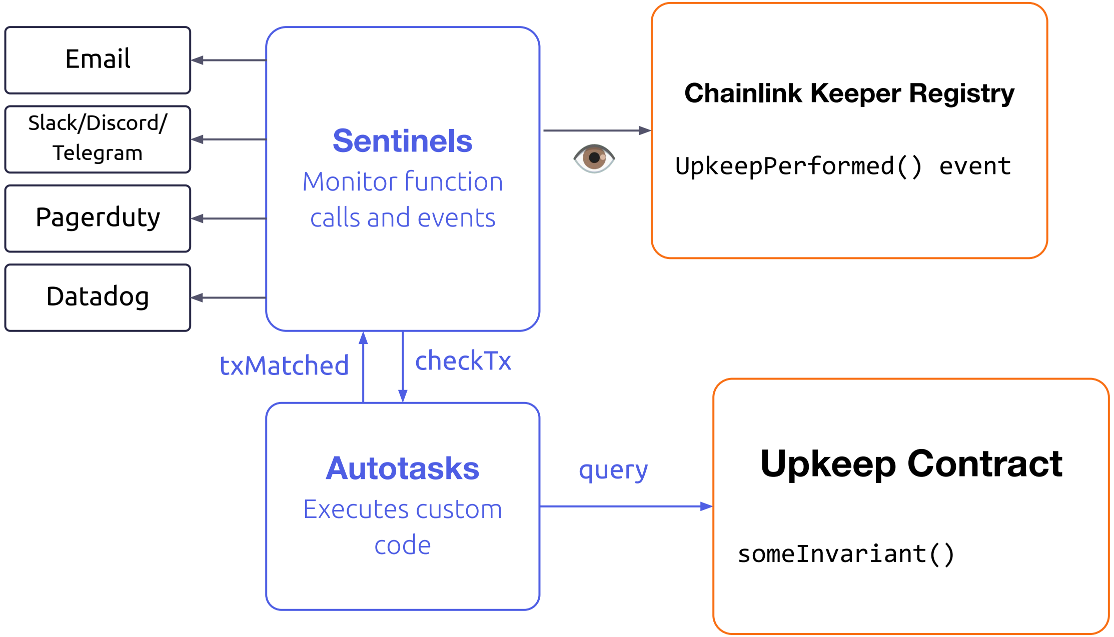
2. Auto funding your Upkeeps



```
const ERC20_ABI = [...];
const REGISTRY_ADDRESS = '0x42dD7716721ba279dA2f1F06F97025d739BD79a8';

const { ethers } = require("ethers");
const { DefenderRelayProvider, DefenderRelaySigner } = require('defender-relay-client/lib/ethers');

const UPKEEP_IDS = ['150', '151'];
const MIN_BALANCE = 5e18.toString();
const REFILL_VALUE = 5e18.toString();
const MAX_VALUE = ethers.BigNumber.from(2).pow(256).sub(1);

exports.handler = async function(credentials) {
  const provider = new DefenderRelayProvider(credentials);
  const signer = new DefenderRelaySigner(credentials, provider, { speed: 'fast' });
  const registry = new ethers.Contract(REGISTRY_ADDRESS, REGISTRY_ABI, signer);

  for (let upkeepId of UPKEEP_IDS) {
    const { balance, target } = await registry.getUpkeep(upkeepId);
    if (balance.lte(MIN_BALANCE)) {
      const token = new ethers.Contract(await registry.LINK(), ERC20_ABI, signer);
      const allowance = await token.allowance(await signer.getAddress(), REGISTRY_ADDRESS);
      if (allowance.lt(REFILL_VALUE)) await token.approve(REGISTRY_ADDRESS, MAX_VALUE).then(tx =>
tx.wait());
      await registry.addFunds(upkeepId, REFILL_VALUE);
    }
  }
}
```

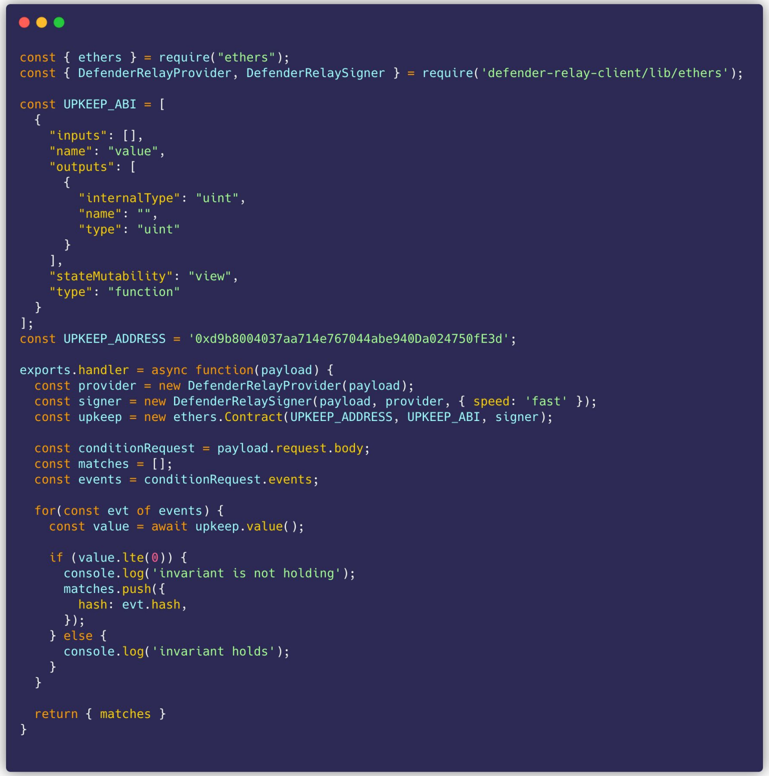
OpenZeppelin

3. Getting failed execution alerts



OpenZeppelin

Getting failed execution alerts



```
const { ethers } = require("ethers");
const { DefenderRelayProvider, DefenderRelaySigner } = require('defender-relay-client/lib/ethers');

const UPKEEP_ABI = [
  {
    "inputs": [],
    "name": "value",
    "outputs": [
      {
        "internalType": "uint",
        "name": "",
        "type": "uint"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  }
];
const UPKEEP_ADDRESS = '0xd9b8004037aa714e767044abe940Da024750fE3d';

exports.handler = async function(payload) {
  const provider = new DefenderRelayProvider(payload);
  const signer = new DefenderRelaySigner(payload, provider, { speed: 'fast' });
  const upkeep = new ethers.Contract(UPKEEP_ADDRESS, UPKEEP_ABI, signer);

  const conditionRequest = payload.request.body;
  const matches = [];
  const events = conditionRequest.events;

  for(const evt of events) {
    const value = await upkeep.value();

    if (value.lte(0)) {
      console.log('invariant is not holding');
      matches.push({
        hash: evt.hash,
      });
    } else {
      console.log('invariant holds');
    }
  }

  return { matches }
}
```

Thank you!

Learn more

openzeppelin.com/**defender**

forum.openzeppelin.com

docs.openzeppelin.com/**defender**

keeper.chain.link

docs.chain.link/docs/**chainlink-keepers**/introduction/