Ethereum Automation

Clara Chick, York Chen, Jan Garong

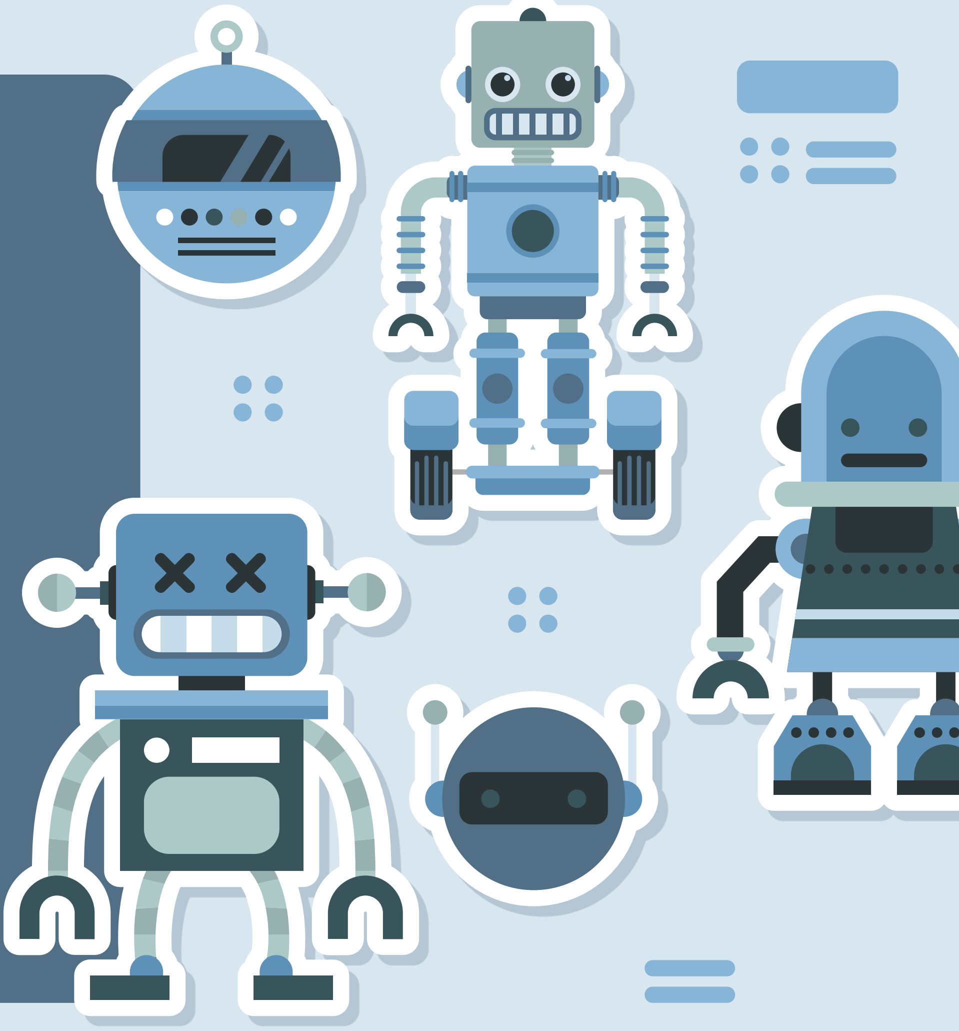
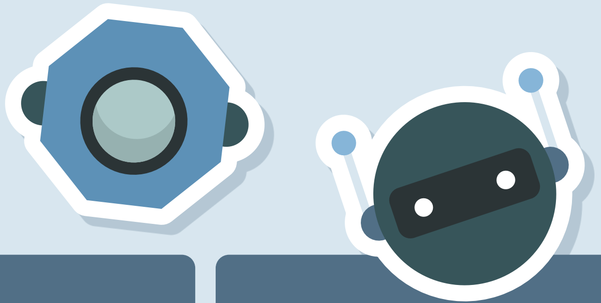
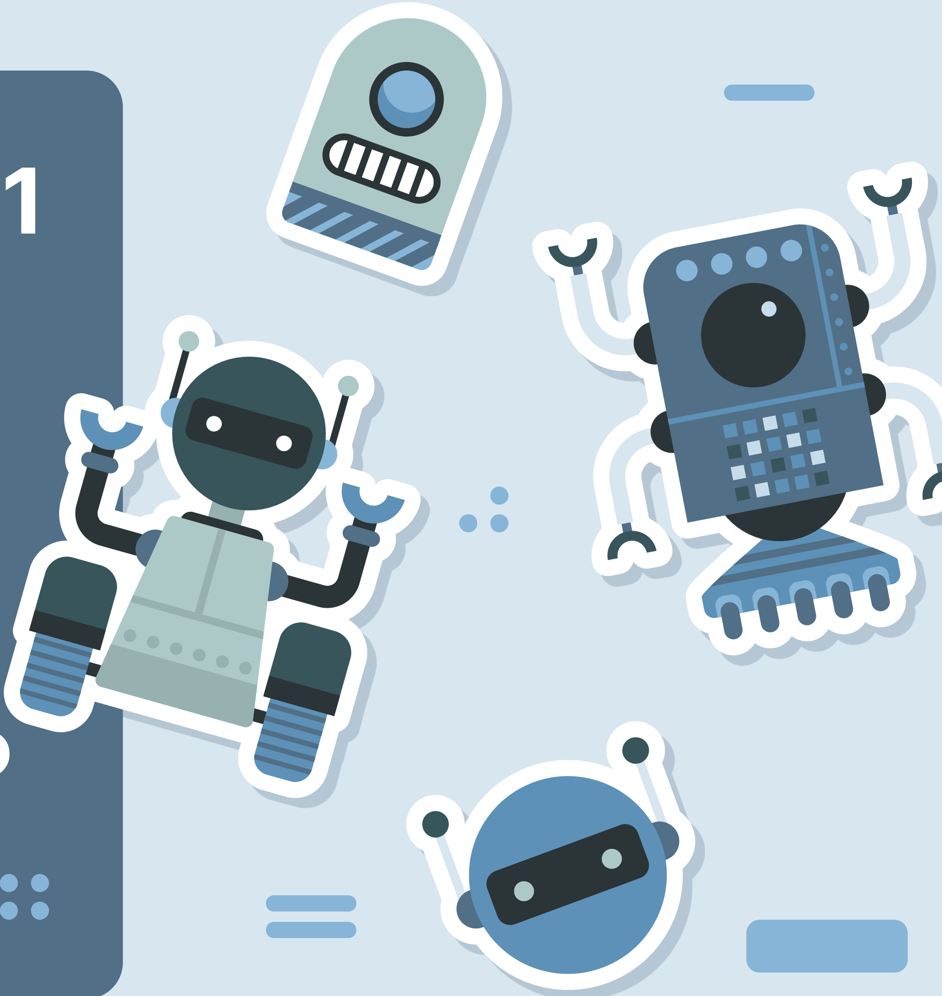


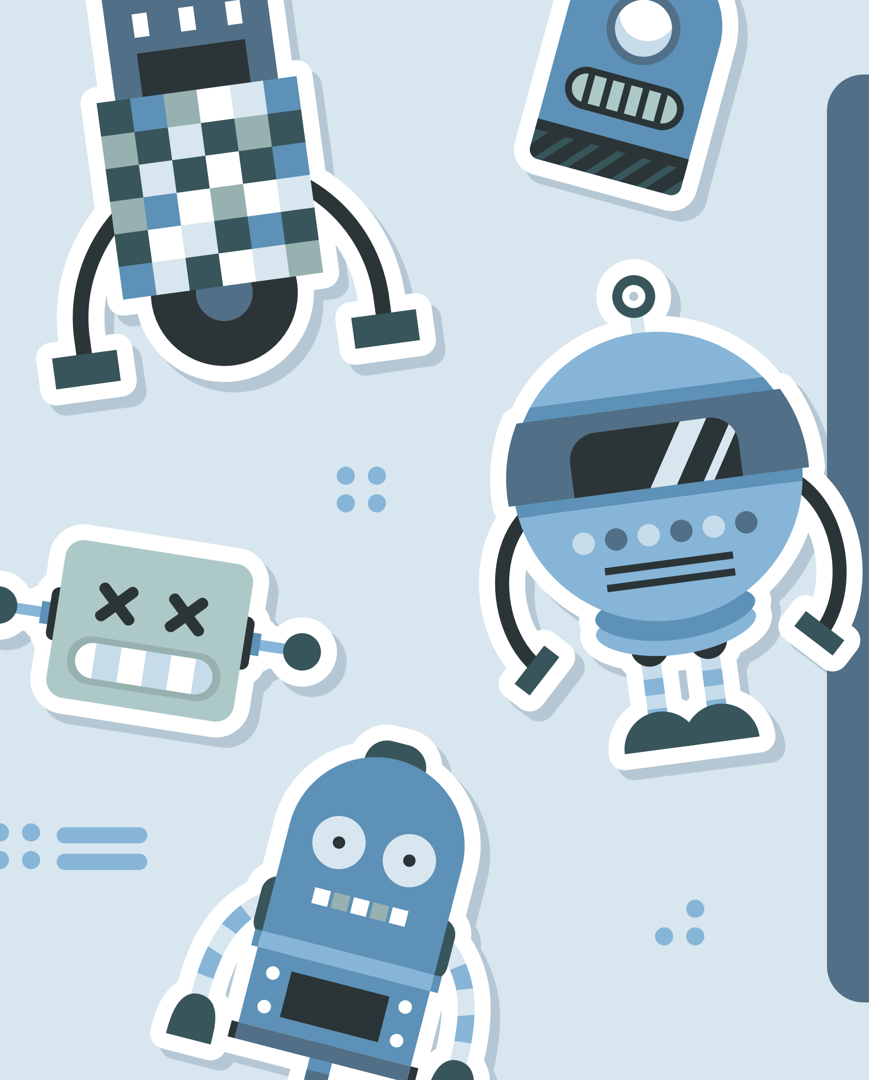
Table of contents



01

What is Ethereum Automation?





Ethereum Automation

It is a process which automates smart contract transactions on EVM-based blockchains

Ideally, it should be able to interact with contracts on the blockchain and reacting to events on the blockchain

Notable Vocabulary

Triggers

Conditions that cause an action to happen.

e.g. Events

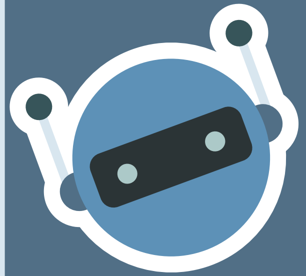
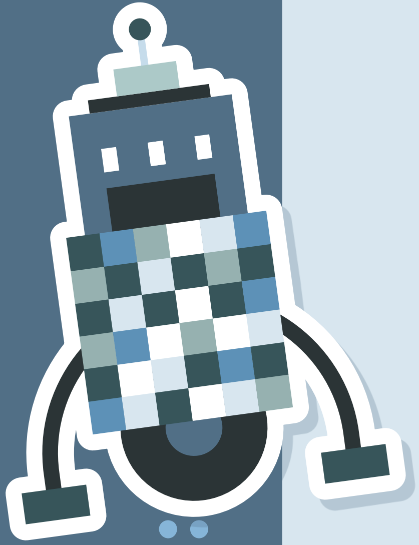
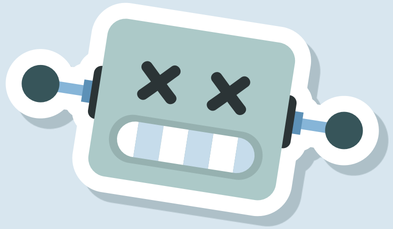
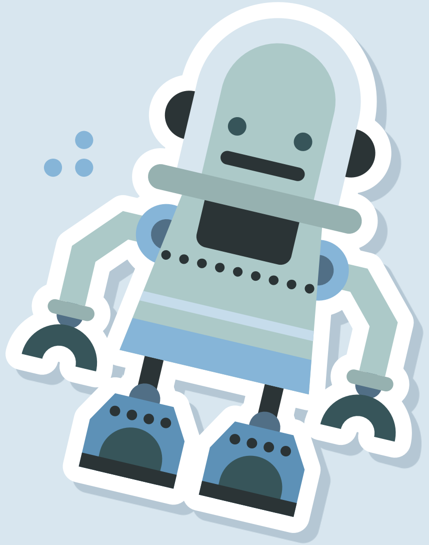
Actions

Side effects of a trigger.

e.g. Sending a transaction

02

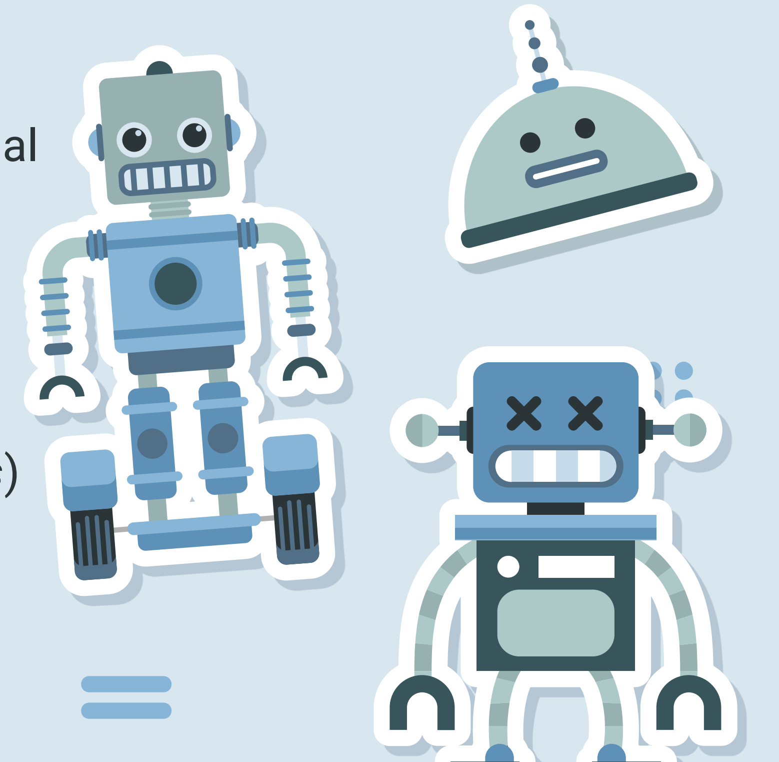
the Why

Why do we need automation?

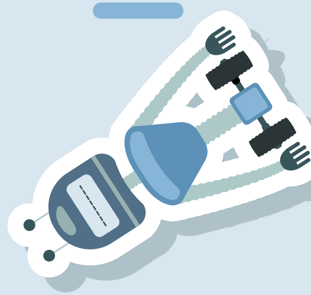
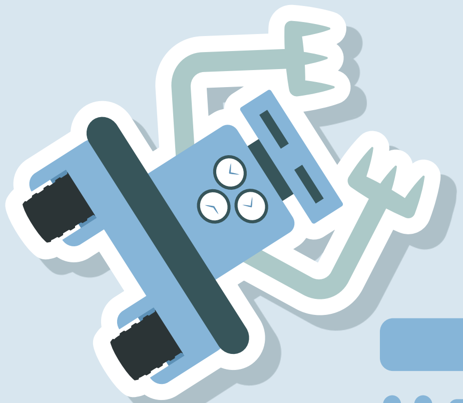
- The need for automation is not exclusive to the Ethereum blockchain
- Smart contracts are functionally inactive if nothing interacts with it
- Interacting with the blockchain is very manual
    - Not practical for developers or users of the smart contract
- We need automation!

- Applications include:
    - Liquidity Management (*e.g. stablecoins*)
    - Automated trading (*e.g. limit orders*)
    - Bridges
    - NFTs



03

Current Solutions

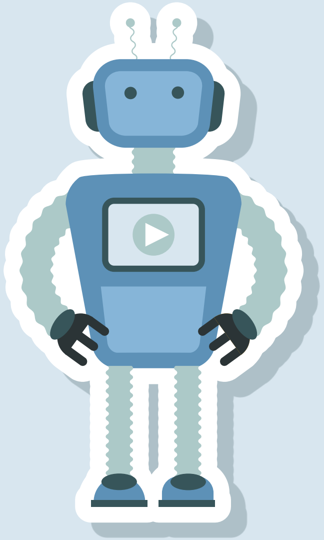
Current Solutions

- Gelato
- Forta
- Chainlink
- Ethereum Alarm Clock
- OpenZeppelin Autotasks

Gelato

https://www.gelato.network/

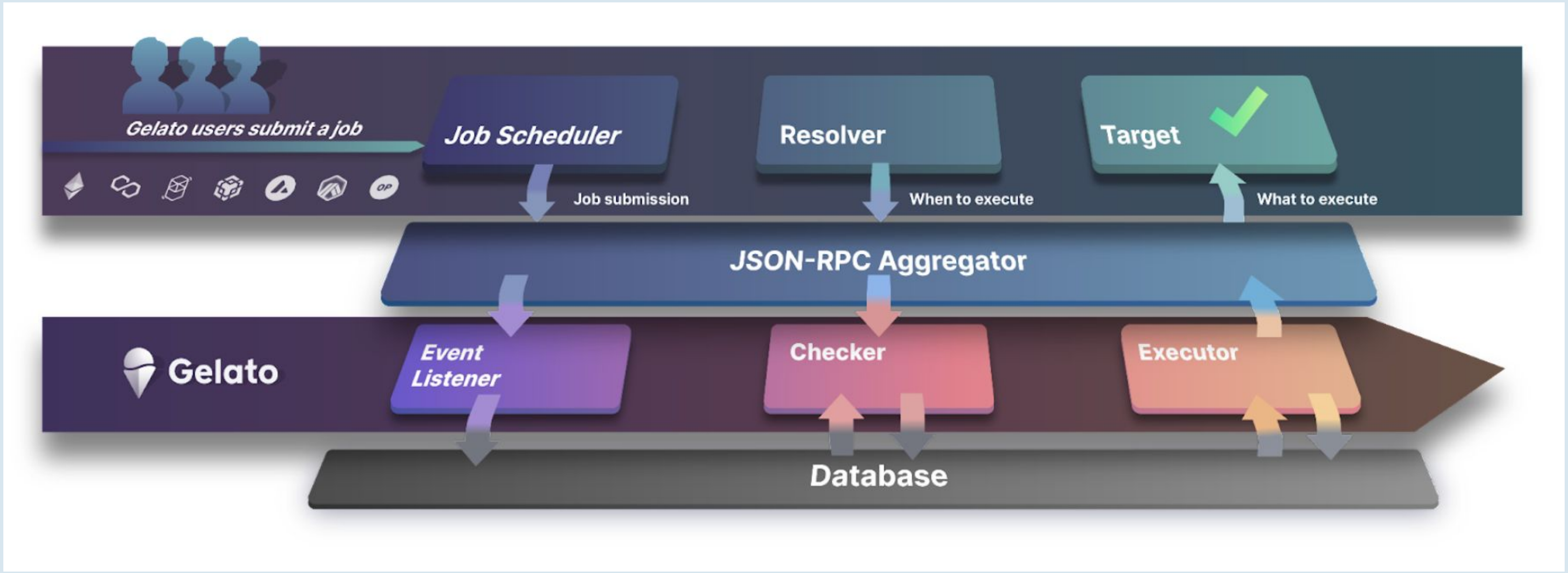
- **Types of Triggers**
  - “Event Listener”
    - Event listener that continuously queries the chain and listens to events
  - “Checker”
    - Custom logic to check
- **Types of Actions**
  - “Executor”
    - Submits transactions on chain
    - Must be funded



Gelato

https://www.gelato.network/

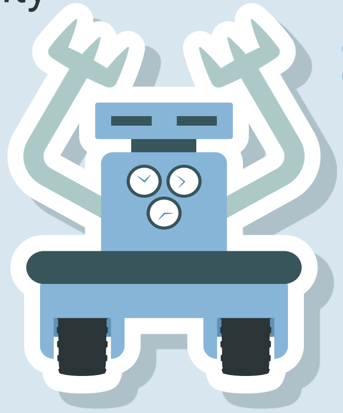
Pros	**Cons**
• Easy to set up (no coding needed) • No need to self host • Gasless transactions	• User interacts with upgradeable proxy: double edged sword, code mutability for centralization!


Gelato users submit a job
Job Scheduler
Resolver
Target
Job submission
When to execute
What to execute
JSON-RPC Aggregator
Gelato
Event Listener
Checker
Executor
Database

Forta

https://forta.org/

- Security and operational monitoring
- **Types of Triggers**
    - Forta detection bots scans the blockchain for security related threats in real-time
        - Can be customized for purposes outside of security
- **Types of Actions**
    - Subscribe to alerts created by bots



Forta

https://forta.org/

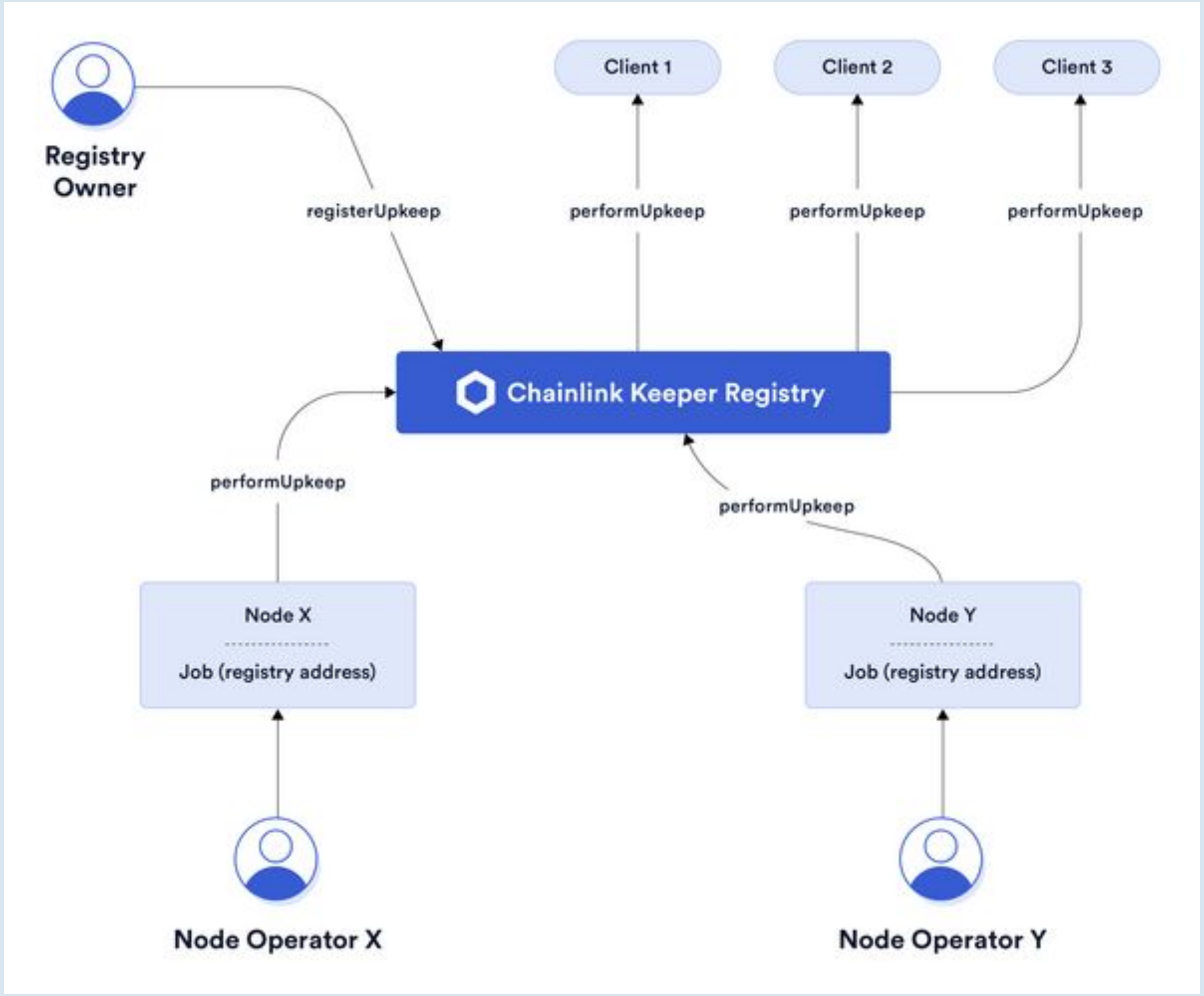
Pros	Cons
● Create a bot both via UI or SDK ● Nodes are incentivized to provide reliable data (or get stake slashed) ● Scan nodes are decentralized	● Requires server resources ● Staking done over Polygon

Chainlink

https://chain.link/automation

- **Types of Triggers**
  - Time-based trigger
  - Custom logic trigger
    - Must be able to be evaluated off-chain
    - *e.g. checking balance on a contract*

- **Types of Actions**
  - Execute smart contract functions


Registry Owner
Client 1
Client 2
Client 3
registerUpkeep
performUpkeep
performUpkeep
performUpkeep
Chainlink Keeper Registry
performUpkeep
performUpkeep
Node X
Job (registry address)
Node Y
Job (registry address)
Node Operator X
Node Operator Y

Chainlink

https://chain.link/automation

Pros	Cons
• Fully decentralized on the Chainlink Network • Supports a variety of EVM networks (Ethereum, Avalanche, Polygon, BNB, etc.) • Only need to pay execution fee	• Can automate/execute only if dApp is integrated with Chainlink contracts

Ethereum Alarm Clock

https://ethereum-alarm-clock.readthedocs.io/en/latest/

- **Types of Triggers**
  - Time-based trigger
    - Created by creating a TransactionRequest smart contract with the action inside.

- **Types of Actions**
  - Execute the task in the smart contract (Solidity) at a given time
    - If done correctly, the caller will receive a payout (gas for execution + fee incentive)

Ethereum Alarm Clock

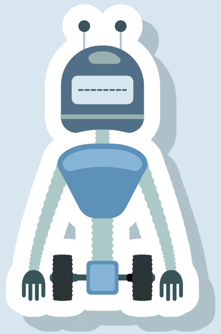
Pros	Cons
• No reliance on smaller blockchains - Just smart contracts on the desired blockchain! • Any EOA can become an executor!	• Lack of punishment for not executing the scheduled transaction

OpenZeppelin Autotasks

https://docs.openzeppelin.com/defender/autotasks

- **Types of Triggers**
  - "Schedule"
    - Time-based trigger
  - "Webhook"
    - Creates a secret URL for webhook providers (i.e. Telegram, Discord)

- **Types of Actions**
  - "Handler Functions"
    - Arbitrary code snippets
    - Relayer integration



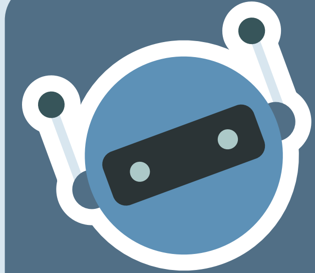
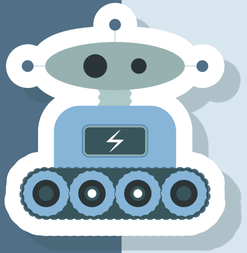
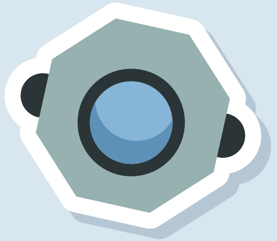
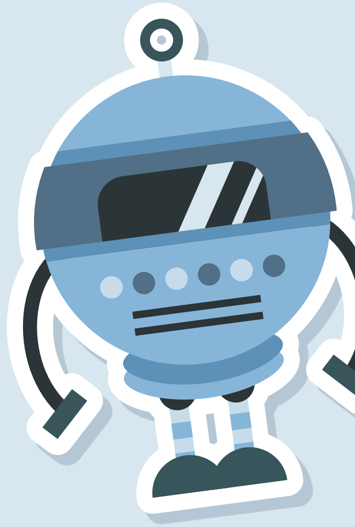
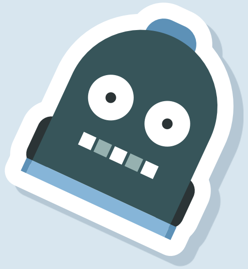
OpenZeppelin Autotasks

https://docs.openzeppelin.com/defender/autotasks

Pros	**Cons**
• Executes JS Functions • No hosting required • Supports off-chain webhooks (Discord, Telegram, etc.)	• Centralized services supports AutoTask: ○ OpenZeppelin Defender (SecOps platform) ○ Amazon Lambda

04

Our Implementation

Terminology of David-bot

Event

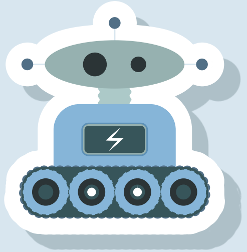
Synonymous to **Trigger**

Task

Synonymous to **Action**

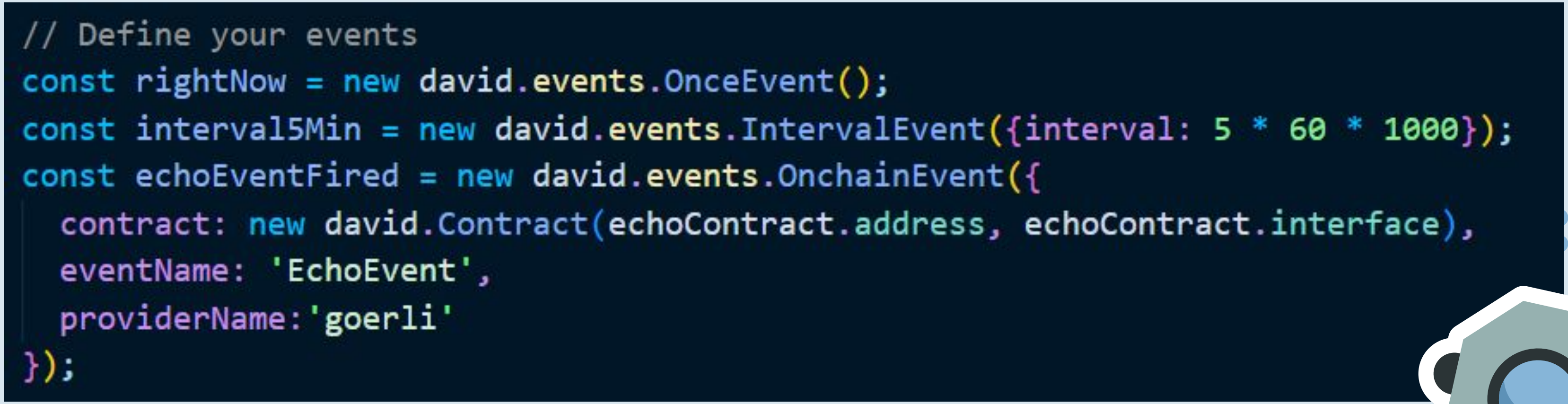
David-bot

- `npm` package which will be used by developers (users) to make their own Ethereum bot(s)
- **Types of Triggers**
  - Time-based trigger (cron)
  - Event-based trigger
    - Configured to be triggered on an event of a specific contract
    - Developers can still implement a version of Chainlink's "custom logic trigger"
- **Types of Actions**
  - Arbitrary Javascript code snippets
  - Relayer integration
    - Execute transactions on the Ethereum blockchain



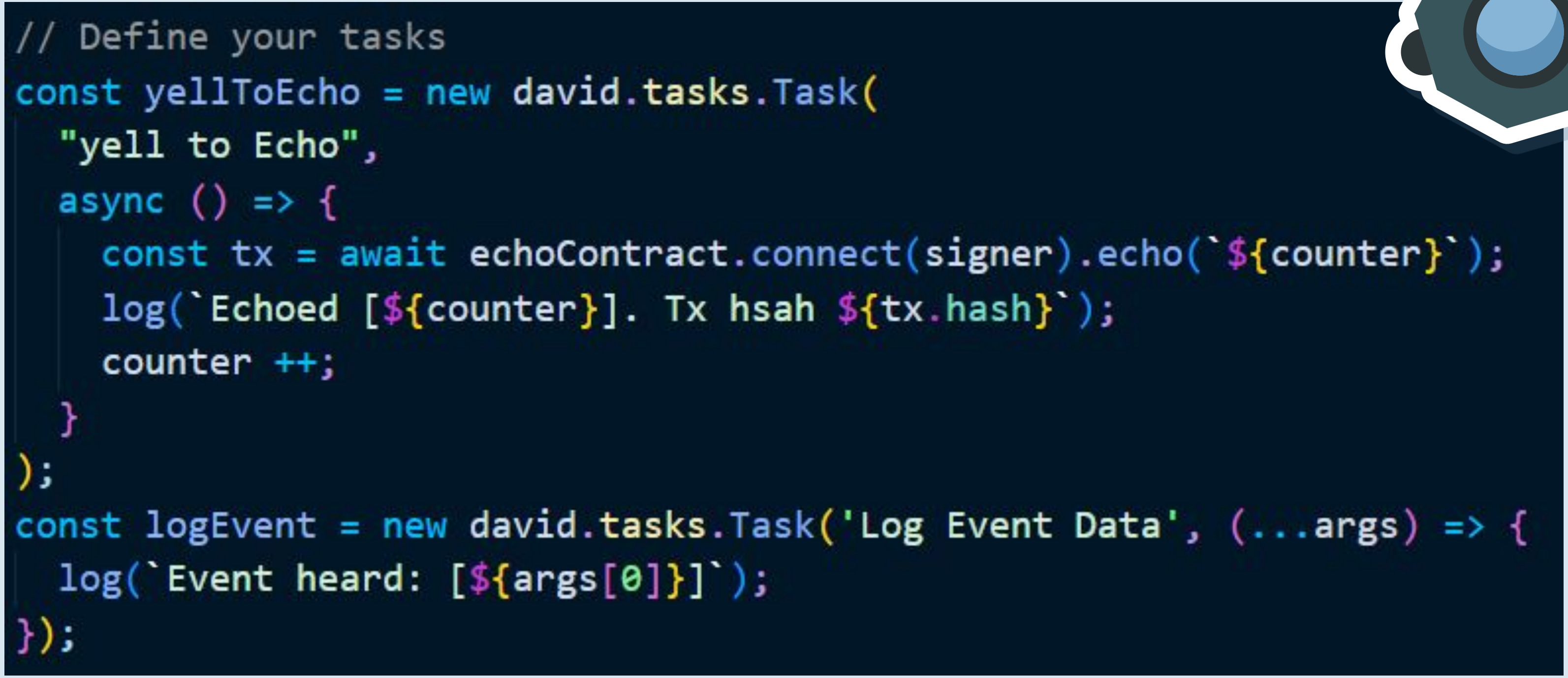
It’s easy!

- To install: `npm i david-bot`
- Import into your project: `import david from ‘david-bot’;`



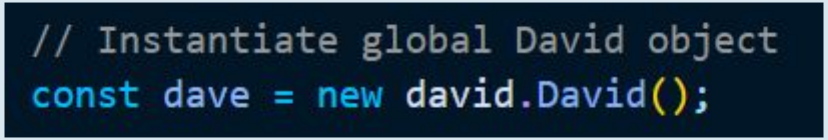
```
// Define your events
const rightNow = new david.events.OnceEvent();
const interval5Min = new david.events.IntervalEvent({interval: 5 * 60 * 1000});
const echoEventFired = new david.events.OnchainEvent({
  contract: new david.Contract(echoContract.address, echoContract.interface),
  eventName: 'EchoEvent',
  providerName:'goerli'
});
```

It’s easy!

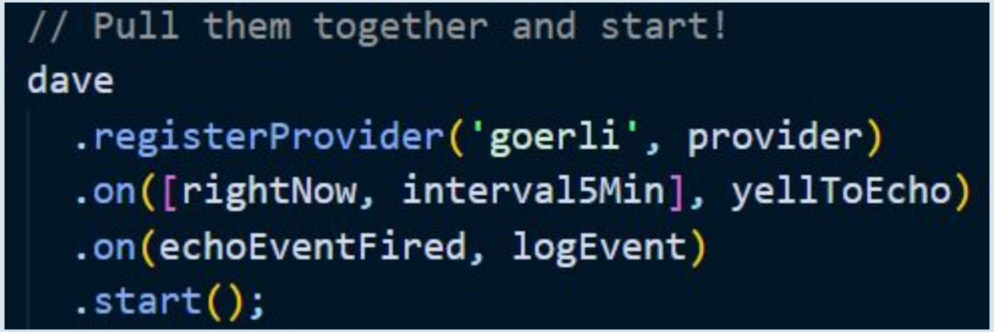


```
// Define your tasks
const yellToEcho = new david.tasks.Task(
  "yell to Echo",
  async () => {
    const tx = await echoContract.connect(signer).echo(`${counter}`);
    log(`Echoed [${counter}]. Tx hsah ${tx.hash}`);
    counter ++;
  }
);
const logEvent = new david.tasks.Task('Log Event Data', (...args) => {
  log(`Event heard: [${args[0]}]`);
});
```

It's easy!

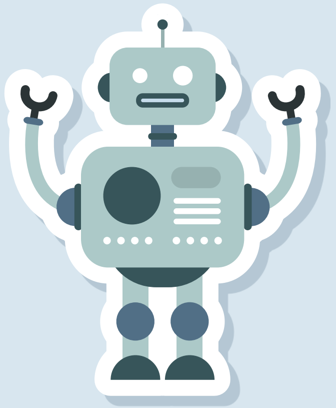


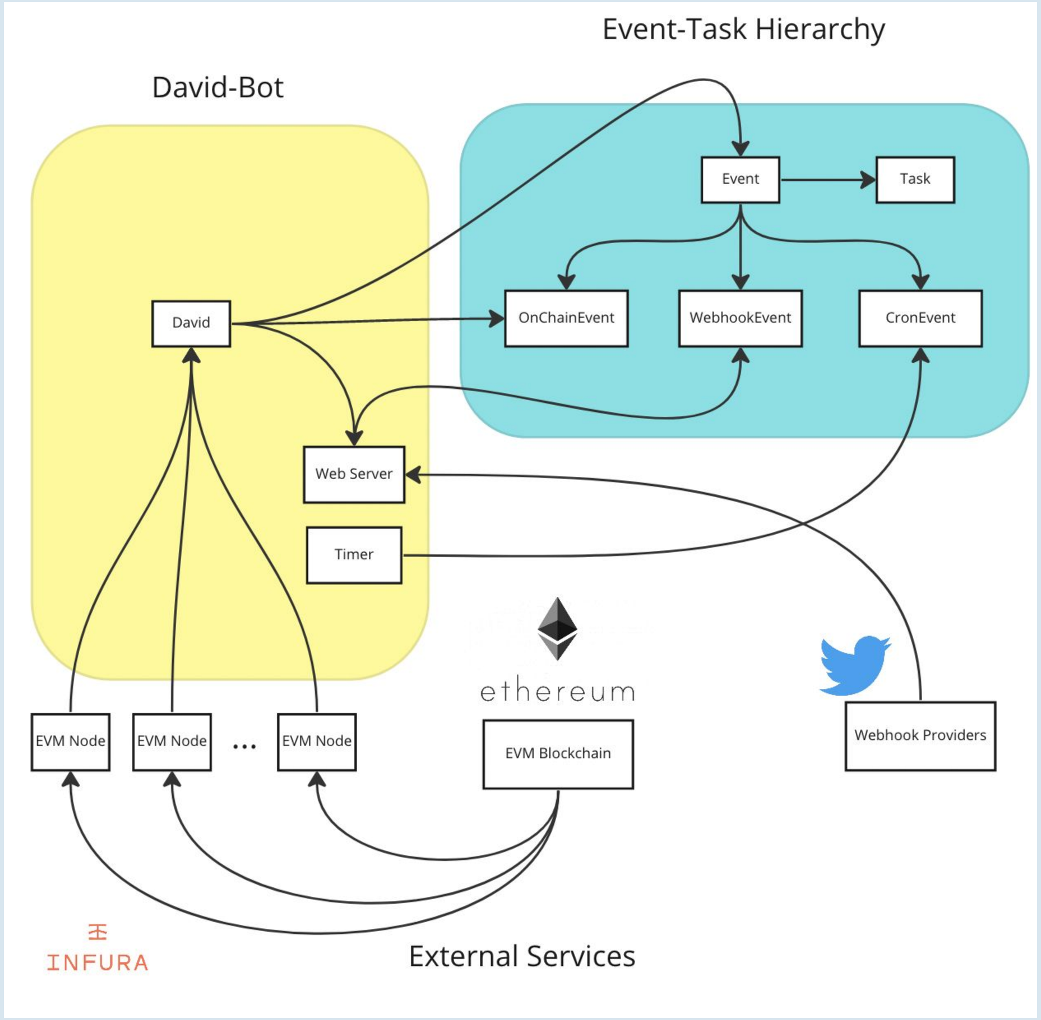
```
// Instantiate global David object
const dave = new david.David();
```



```
// Pull them together and start!
dave
  .registerProvider('goerli', provider)
  .on([rightNow, interval5Min], yellToEcho)
  .on(echoEventFired, logEvent)
  .start();
```

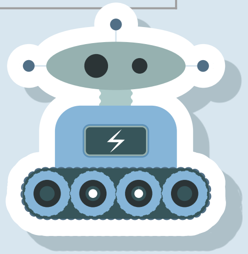
hurray. i have
been given
life.

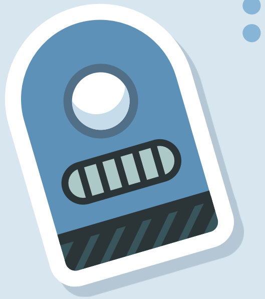
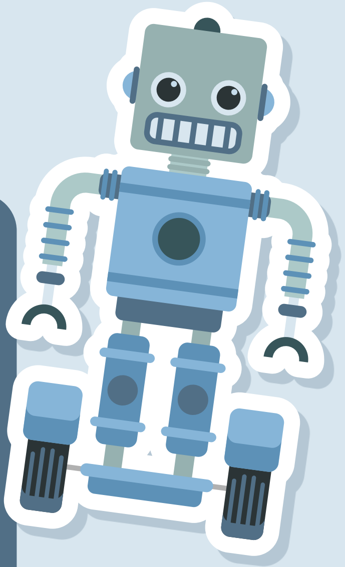



Event-Task Hierarchy
David-Bot
Event
Task
David
OnChainEvent
WebhookEvent
CronEvent
Web Server
Timer
ethereum
EVM Node
EVM Node
...
EVM Node
EVM Blockchain
Webhook Providers
INFURA
External Services

David-bot

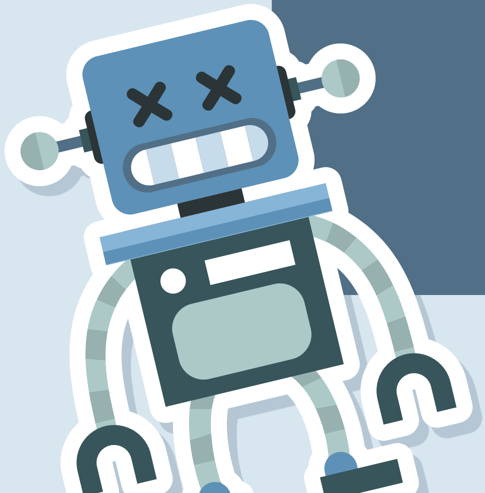
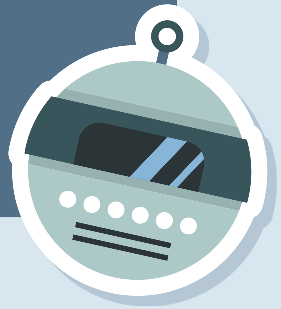
Pros	Cons
● Self-hosted ● Can execute arbitrary JS code ● Supports off-chain web hooks ● Ensures resiliency by using multiple providers	● Dependant on Ethers ● Need to provide their own provider(s)

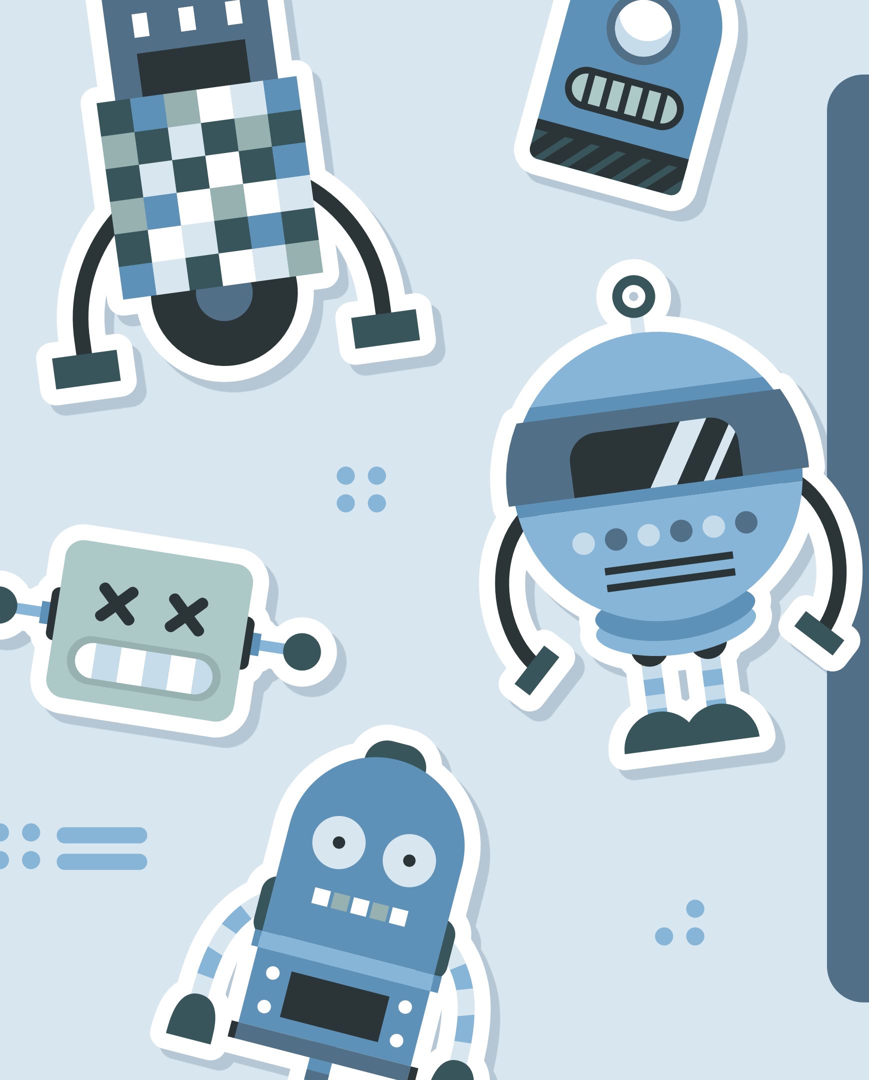


05

demo



Thank you for listening!

Q&A