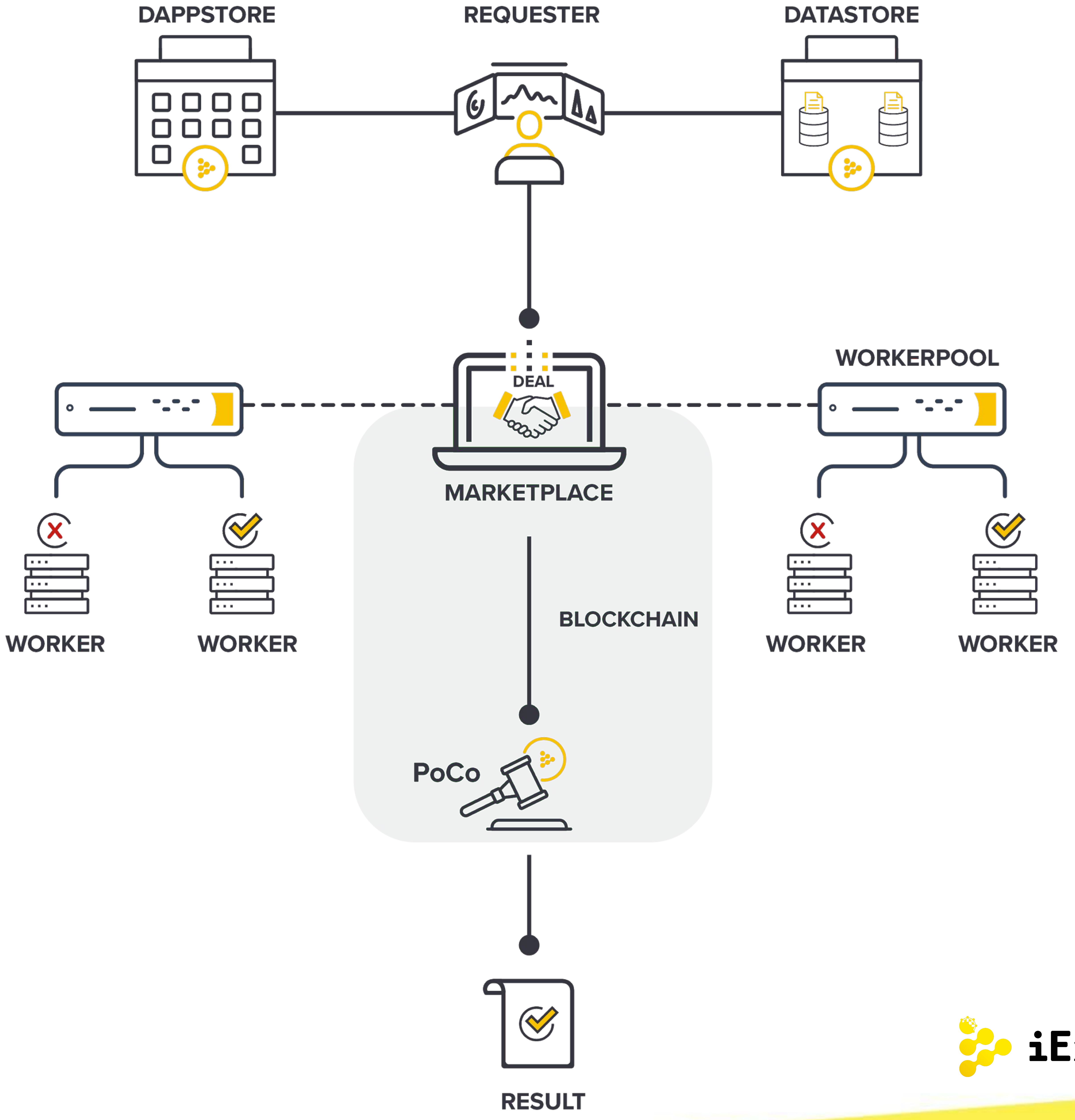
Talk transaction of iExec research
From : sidechains and bridges
To : interoperable substrate chains

Francois Branciard
fb@iex.ec
@fbranciard
www.iex.ec

iExec ecosystem



iExec

Off chain compute

iExec ecosystem

- Only basic algorithms can be reasonably run in EVM
- Off chain compute to extend capacity of dapp
- Proof of Contribution : Economic games to preserve result trust *
- Marketplace resources : datasets, applications, servers providers

POCO

* https://docs.iex.ec/poco.html

Sidechain strategy

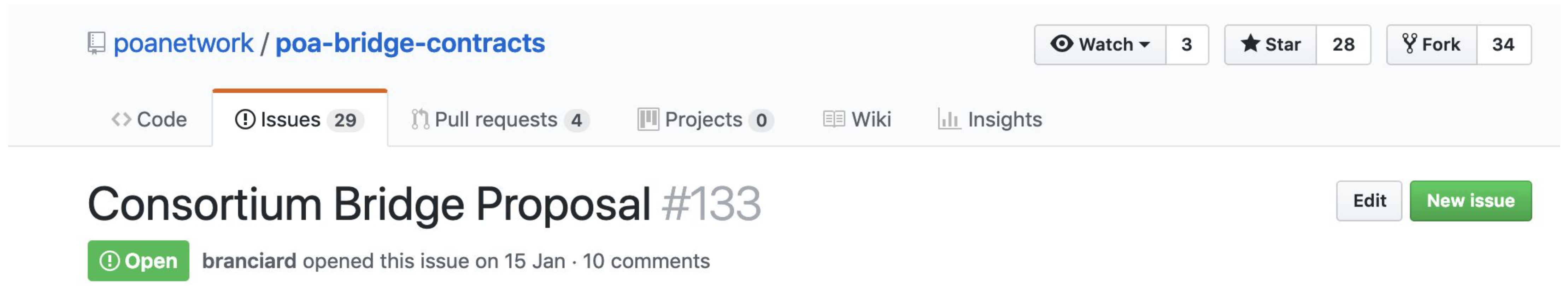
iExec ecosystem in a web3 perspective

- **From : Deliver** pragmatic intermediary solution with existing technologies:
  - Mainnet EVM smart contract
  - PoA chain EVM smart contract
  - Bridges (EVM <-> EVM)
  - Mitigate governance with consortium
- **To : Research** on an optimal solution :
  - Replace bridges by dedicated relay-chain, with their own incentives, to link messages between chains. aka polkadot
  - Governance modules : block production, network upgrades. aka substrate modules
  - Owned autonomous domain chain incentive and governance

Delivery in progress : Consortium bridge

Consortium bridge

- ERC-20 ERC-20 Token bridge **Poa-network**
  Adding **Consortium Bridge feature** on Poa-network bridge
  - use Mainnet token asset in consortium
  - limit bridge responsibility for whitelisted addresses



Consortium Bridge Proposal #133

Open branciard opened this issue on 15 Jan · 10 comments

  - https://github.com/poanetwork/poa-bridge-contracts/issues/133
    https://forum.poa.network/t/consortium-bridge/1739

- Illustration of consortium bridge usage

Let’s build a consortium

Consortium bridge

Example of **Application verticals**

ROLE	Data Provider	App Provider	Resource Provider
Expertise	A company that have dataset that can be used to train AI model	A company that provide algorithm to trained model.	A company that has idle GPU resources
Authority	🔑	🔑	🔑

Let’s build a consortium

Consortium bridge

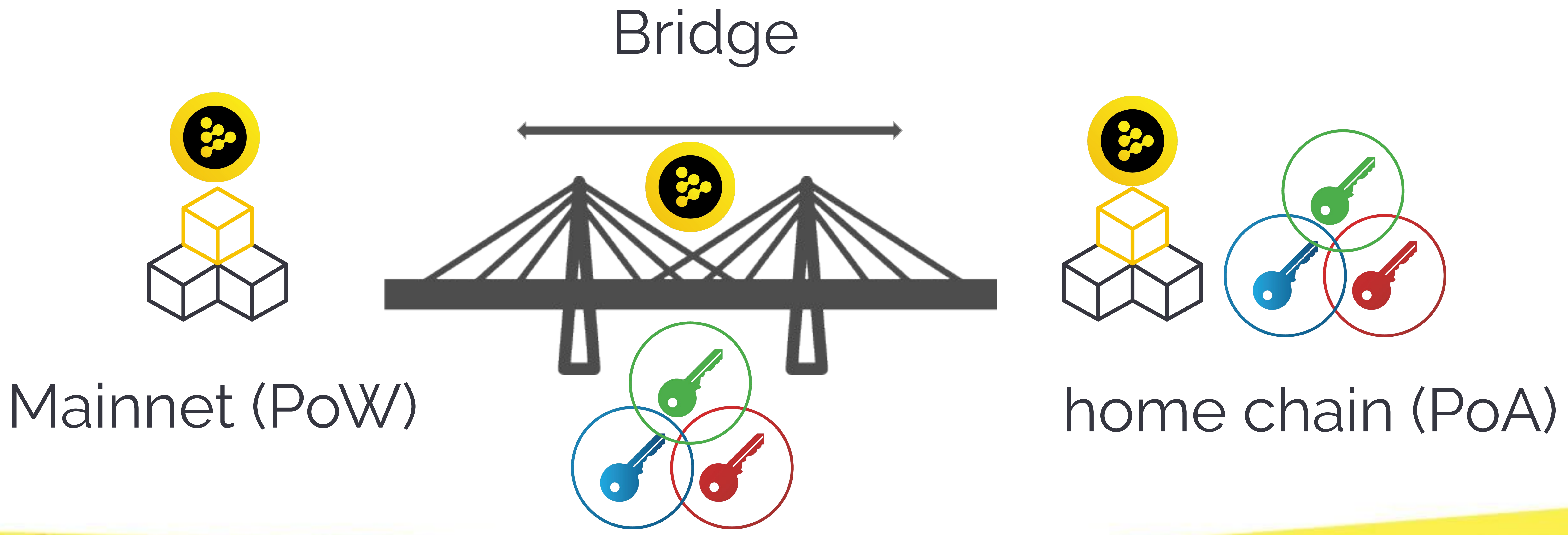
- PoA **chain** under authority :
- Dataset Provider deploys **dataset** with iExec Stack
- Application Provider deploys **app** with iExec Stack
- Resource Provider creates **workerpool** with iExec Stack
- PoCo Transactions : shared **auditability** of **usage** between parties

POCO

Let's automate token payment between them

Consortium bridge

- add a **consortium bridge** under same authority than can **deposit** and **withdraw** between the mainnet and their "home" chain.
- use marketplace contract to set dataset, app, resources dynamic **prices usage between them**.



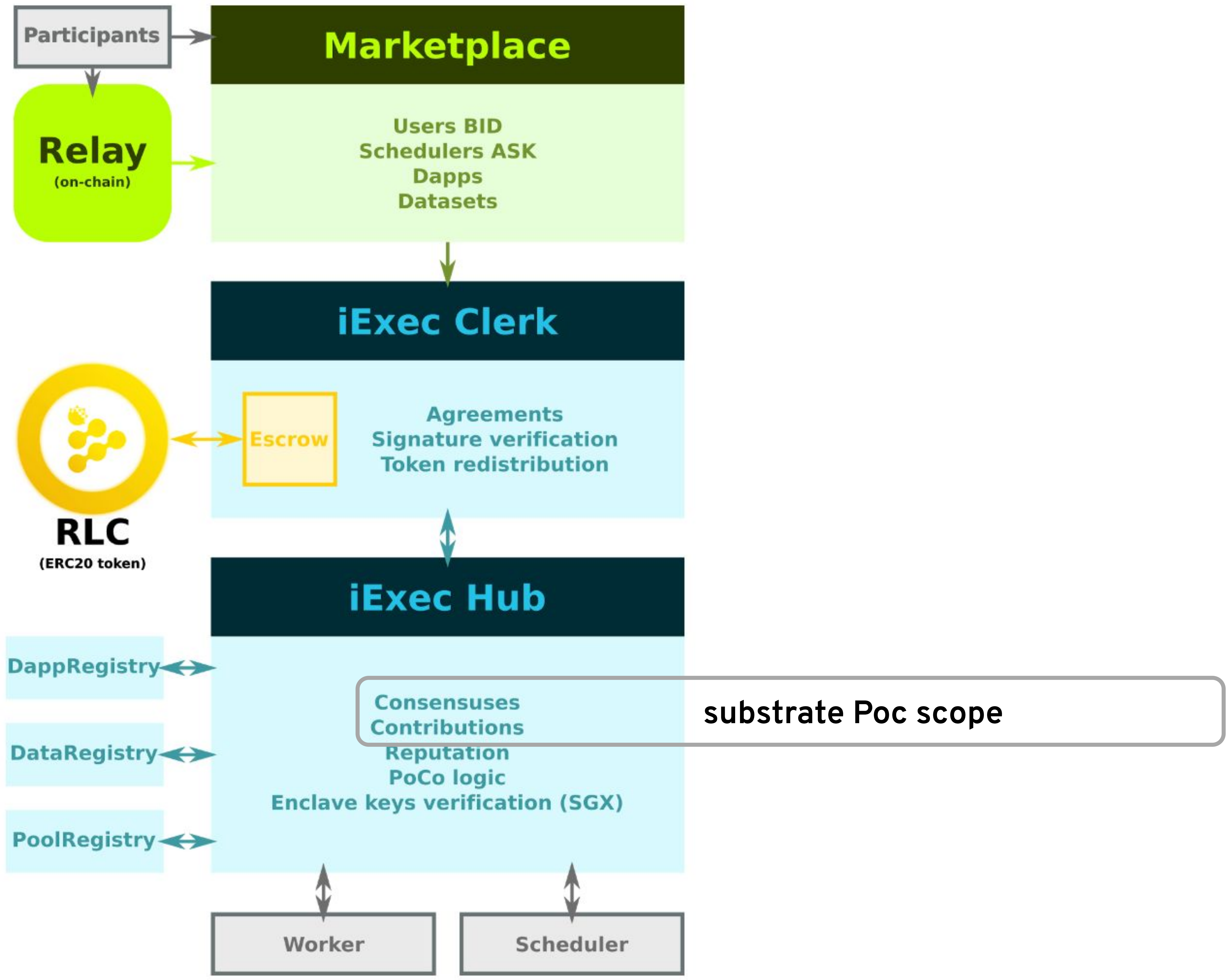
Research in progress : Substrate POC

Substrate POC

- From substrate-template-node providing base modules:
  - System, Timestamp, Consensus, Balances etc ...
- Custom modules can be added, iExec first poc scope :
  - modelize Tasks and workers contributions
  - contribute, reveal schemas with economic games incentive.
- Objective :
  - familiarized with wasm (next blockchain standard to come ?)
  - runtimes vs smart contract feedback
  - explore what on-chain governance modules can offer

First poc scope

Substrate POC



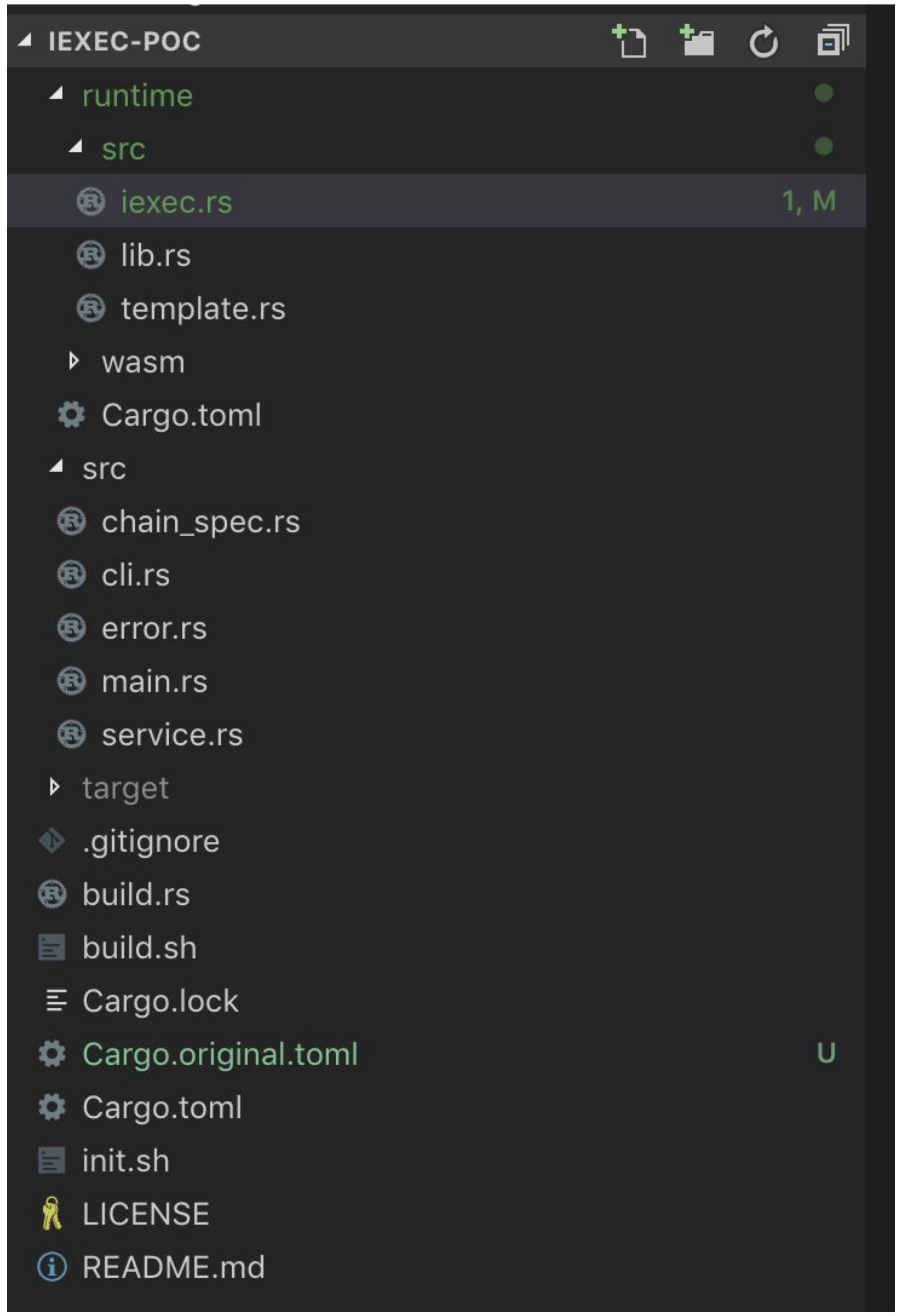
Demo

Substrate POC

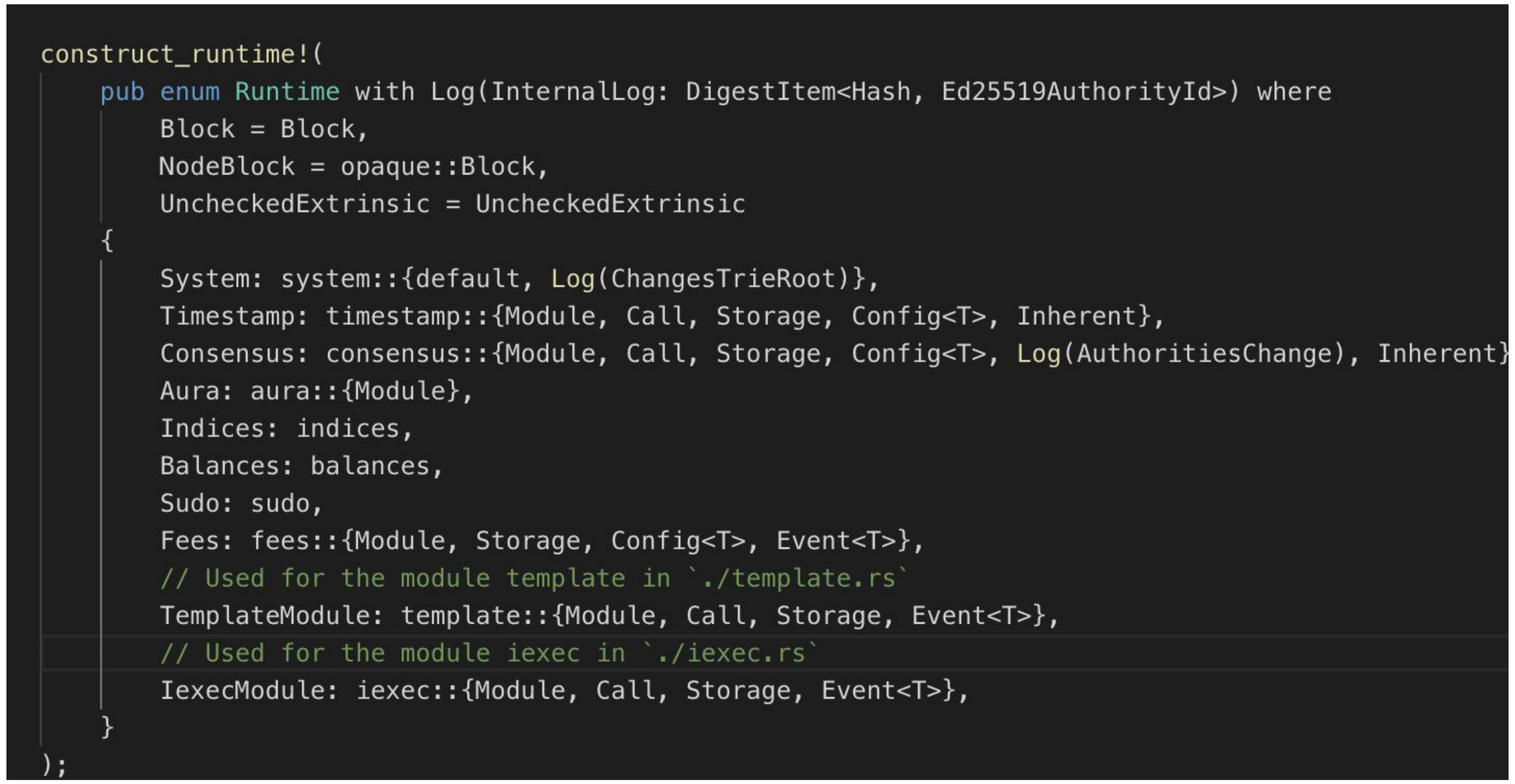
iExec

From substrate node template

Substrate poc



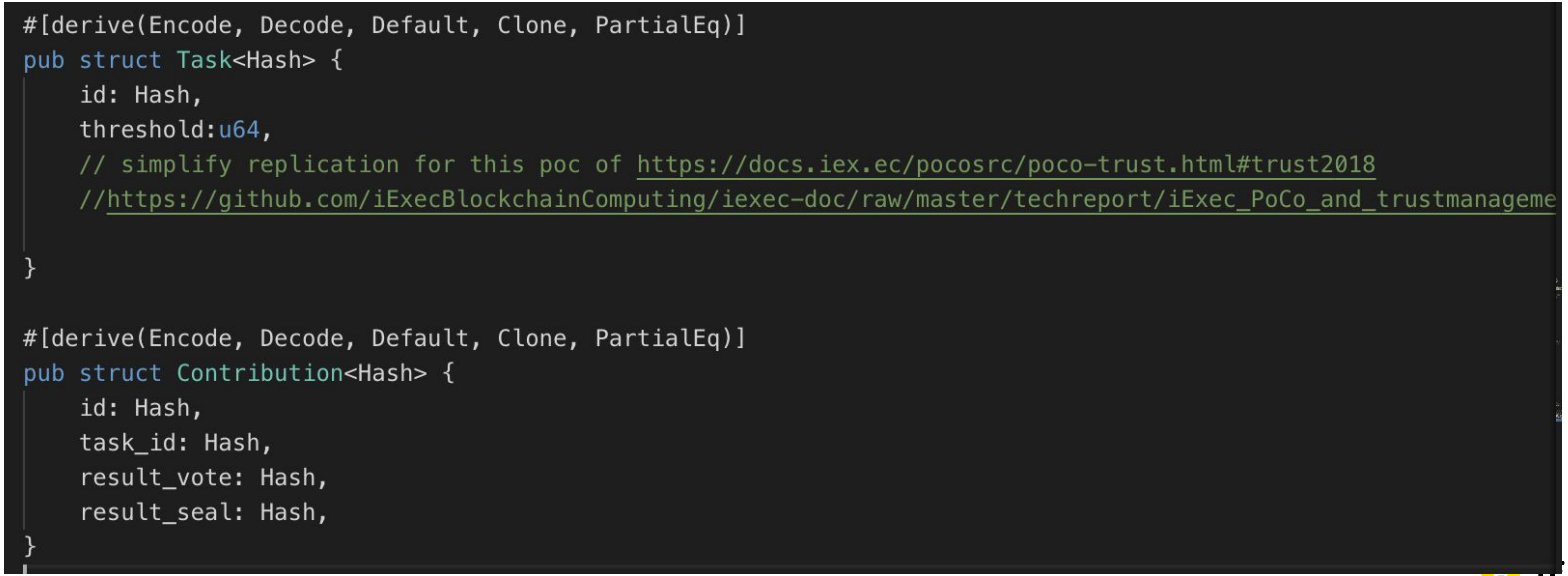
```
IEXEC-POC
  runtime
    src
      iexec.rs        1, M
      lib.rs
      template.rs
    wasm
    Cargo.toml
  src
    chain_spec.rs
    cli.rs
    error.rs
    main.rs
    service.rs
  target
  .gitignore
  build.rs
  build.sh
  Cargo.lock
  Cargo.original.toml   U
  Cargo.toml
  init.sh
  LICENSE
  README.md
```



```
construct_runtime!(
    pub enum Runtime with Log(InternalLog: DigestItem<Hash, Ed25519AuthorityId>) where
        Block = Block,
        NodeBlock = opaque::Block,
        UncheckedExtrinsic = UncheckedExtrinsic
    {
        System: system::{default, Log(ChangesTrieRoot)},
        Timestamp: timestamp::{Module, Call, Storage, Config<T>, Inherent},
        Consensus: consensus::{Module, Call, Storage, Config<T>, Log(AuthoritiesChange), Inherent}
        Aura: aura::{Module},
        Indices: indices,
        Balances: balances,
        Sudo: sudo,
        Fees: fees::{Module, Storage, Config<T>, Event<T>},
        // Used for the module template in `./template.rs`
        TemplateModule: template::{Module, Call, Storage, Event<T>},
        // Used for the module iexec in `./iexec.rs`
        IexecModule: iexec::{Module, Call, Storage, Event<T>},
    }
);
```

Struct Model

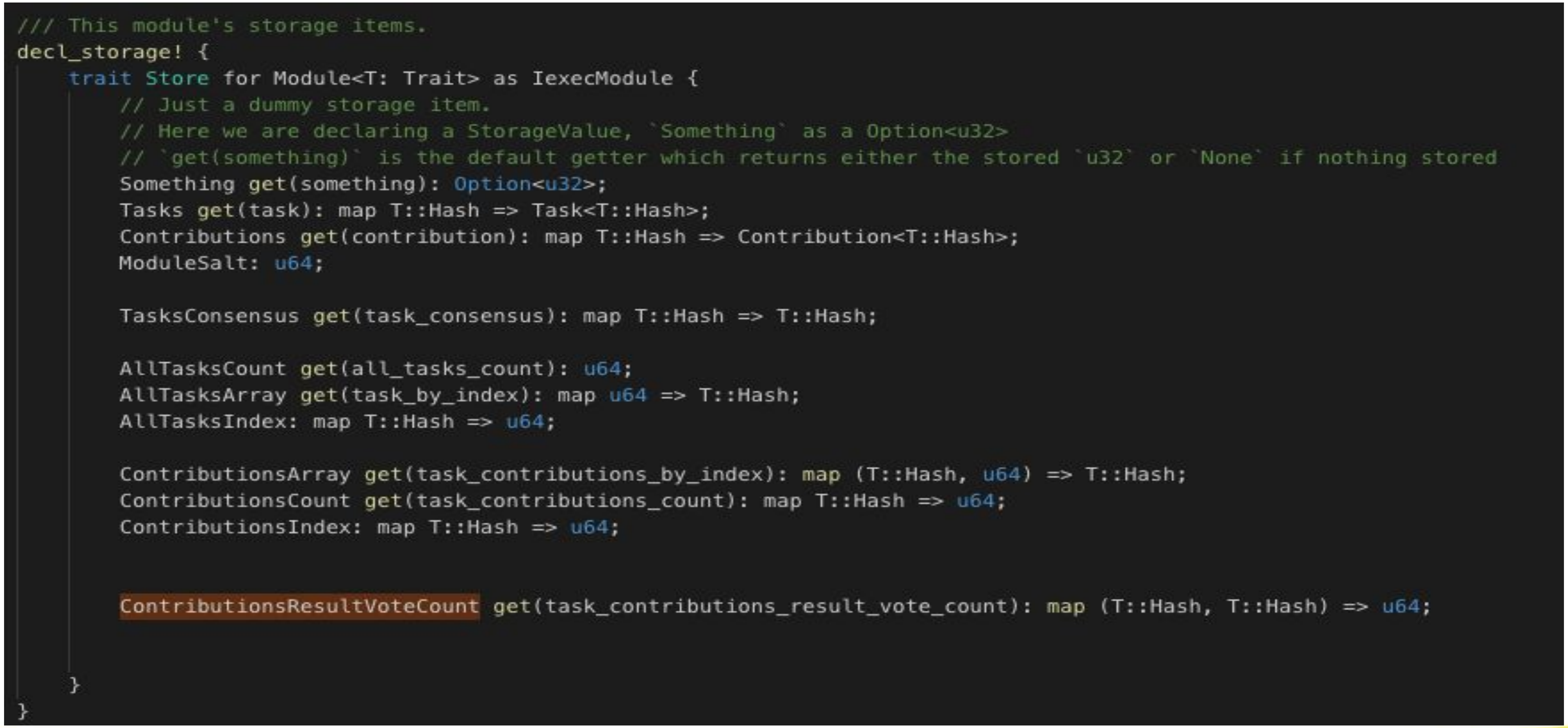
Substrate poc



```
#[derive(Encode, Decode, Default, Clone, PartialEq)]
pub struct Task<Hash> {
    id: Hash,
    threshold:u64,
    // simplify replication for this poc of https://docs.iex.ec/pocosrc/poco-trust.html#trust2018
    //https://github.com/iExecBlockchainComputing/iexec-doc/raw/master/techreport/iExec_PoCo_and_trustmanageme

}

#[derive(Encode, Decode, Default, Clone, PartialEq)]
pub struct Contribution<Hash> {
    id: Hash,
    task_id: Hash,
    result_vote: Hash,
    result_seal: Hash,
}
```

Define Storage

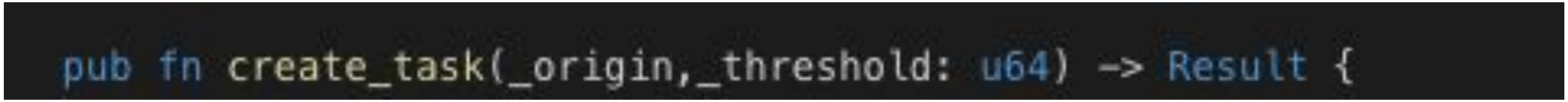
Substrate poc



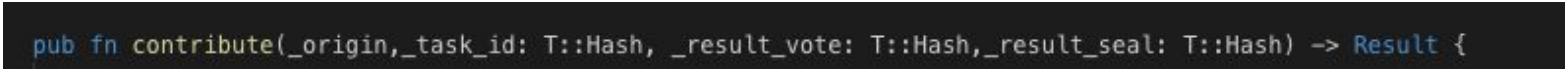
```
/// This module's storage items.
decl_storage! {
    trait Store for Module<T: Trait> as IexecModule {
        // Just a dummy storage item.
        // Here we are declaring a StorageValue, `Something` as a Option<u32>
        // `get(something)` is the default getter which returns either the stored `u32` or `None` if nothing stored
        Something get(something): Option<u32>;
        Tasks get(task): map T::Hash => Task<T::Hash>;
        Contributions get(contribution): map T::Hash => Contribution<T::Hash>;
        ModuleSalt: u64;

        TasksConsensus get(task_consensus): map T::Hash => T::Hash;

        AllTasksCount get(all_tasks_count): u64;
        AllTasksArray get(task_by_index): map u64 => T::Hash;
        AllTasksIndex: map T::Hash => u64;

        ContributionsArray get(task_contributions_by_index): map (T::Hash, u64) => T::Hash;
        ContributionsCount get(task_contributions_count): map T::Hash => u64;
        ContributionsIndex: map T::Hash => u64;


        ContributionsResultVoteCount get(task_contributions_result_vote_count): map (T::Hash, T::Hash) => u64;


    }
}
```

Define functions

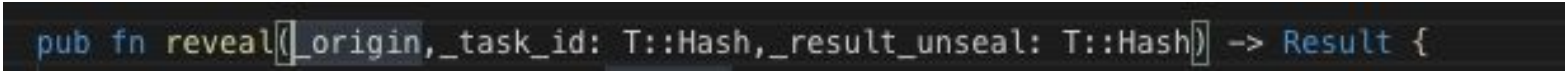
Substrate poc



```
pub fn create_task(_origin,_threshold: u64) -> Result {
```



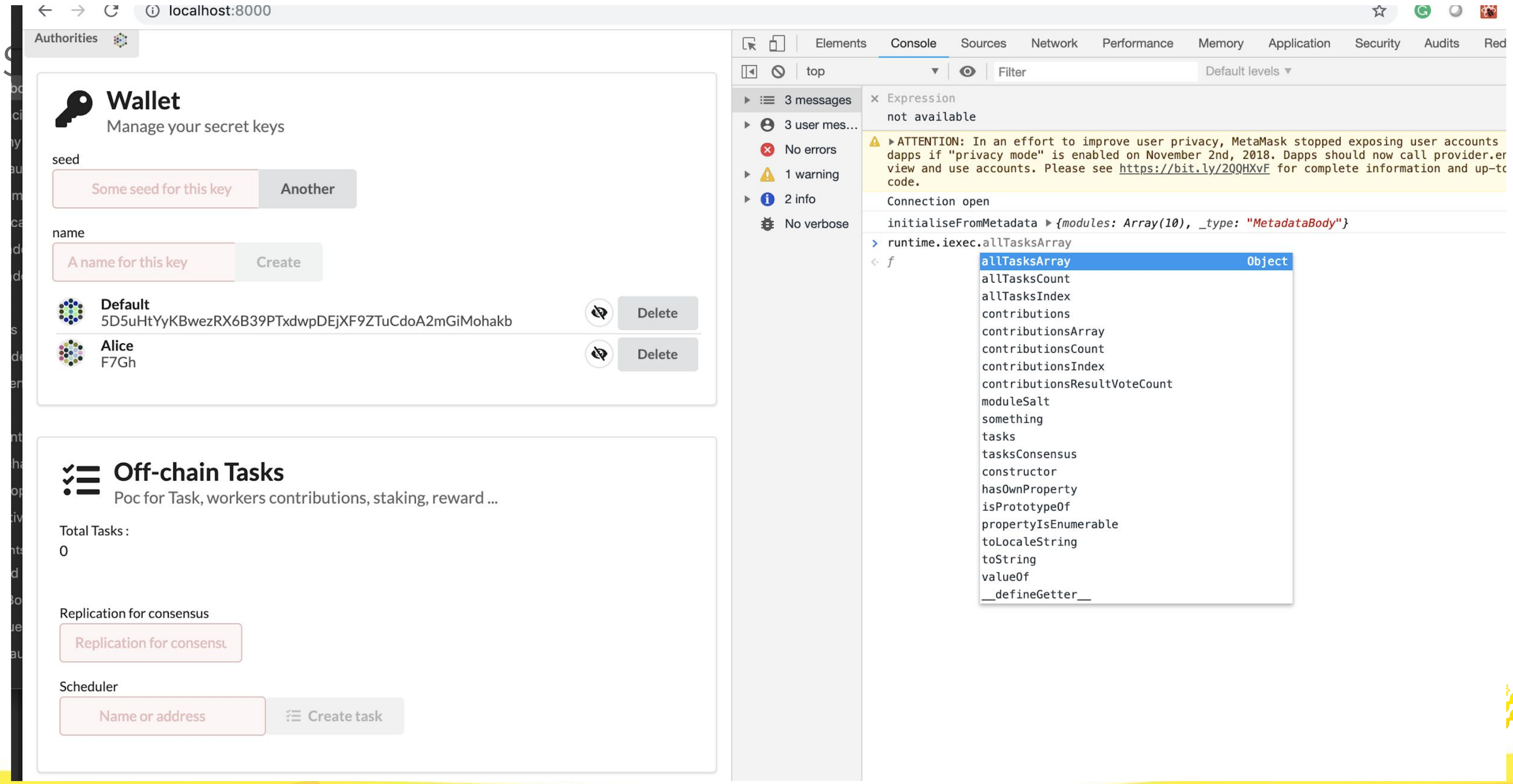
```
pub fn contribute(_origin,_task_id: T::Hash, _result_vote: T::Hash,_result_seal: T::Hash) -> Result {
```



```
pub fn reveal(_origin,_task_id: T::Hash,_result_unseal: T::Hash) -> Result {
```

iExec

Substrate-ui



Wallet
Manage your secret keys

seed: Some seed for this key — Another

name: A name for this key — Create

Default
5D5uHtYyKBwezRX6B39PTxdwpDEjXF9ZTuCdoA2mGiMohakb — Delete

Alice
F7Gh — Delete

Off-chain Tasks
Poc for Task, workers contributions, staking, reward ...

Total Tasks :
0

Replication for consensus: Replication for consensus

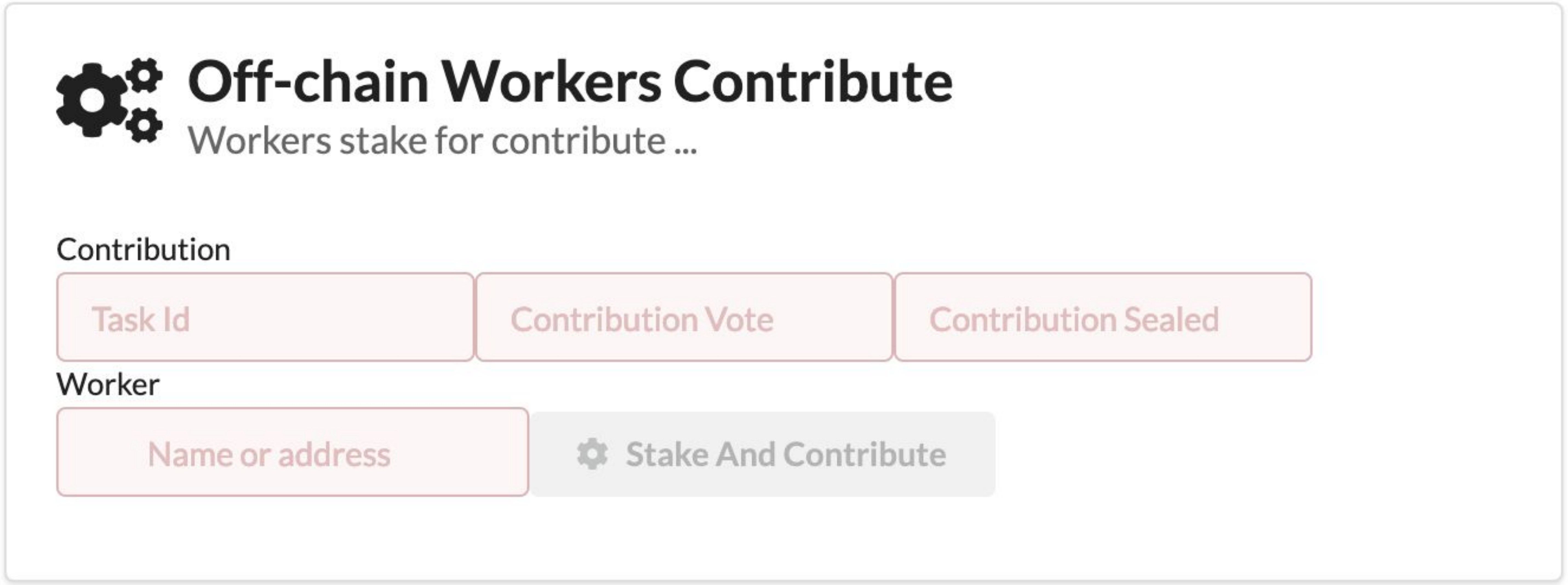
Scheduler: Name or address — Create task

```
Expression
not available
ATTENTION: In an effort to improve user privacy, MetaMask stopped exposing user accounts
dapps if "privacy mode" is enabled on November 2nd, 2018. Dapps should now call provider.er
view and use accounts. Please see https://bit.ly/2QQHXvF for complete information and up-to
code.
Connection open
initialiseFromMetadata ▶{modules: Array(10), _type: "MetadataBody"}
> runtime.iexec.allTasksArray
< f
  allTasksArray                             Object
  allTasksCount
  allTasksIndex
  contributions
  contributionsArray
  contributionsCount
  contributionsIndex
  contributionsResultVoteCount
  moduleSalt
  something
  tasks
  tasksConsensus
  constructor
  hasOwnProperty
  isPrototypeOf
  propertyIsEnumerable
  toLocaleString
  toString
  valueOf
  __defineGetter__
```

iExec

Contribution reveal

Substrate poc



Off-chain Workers Contribute

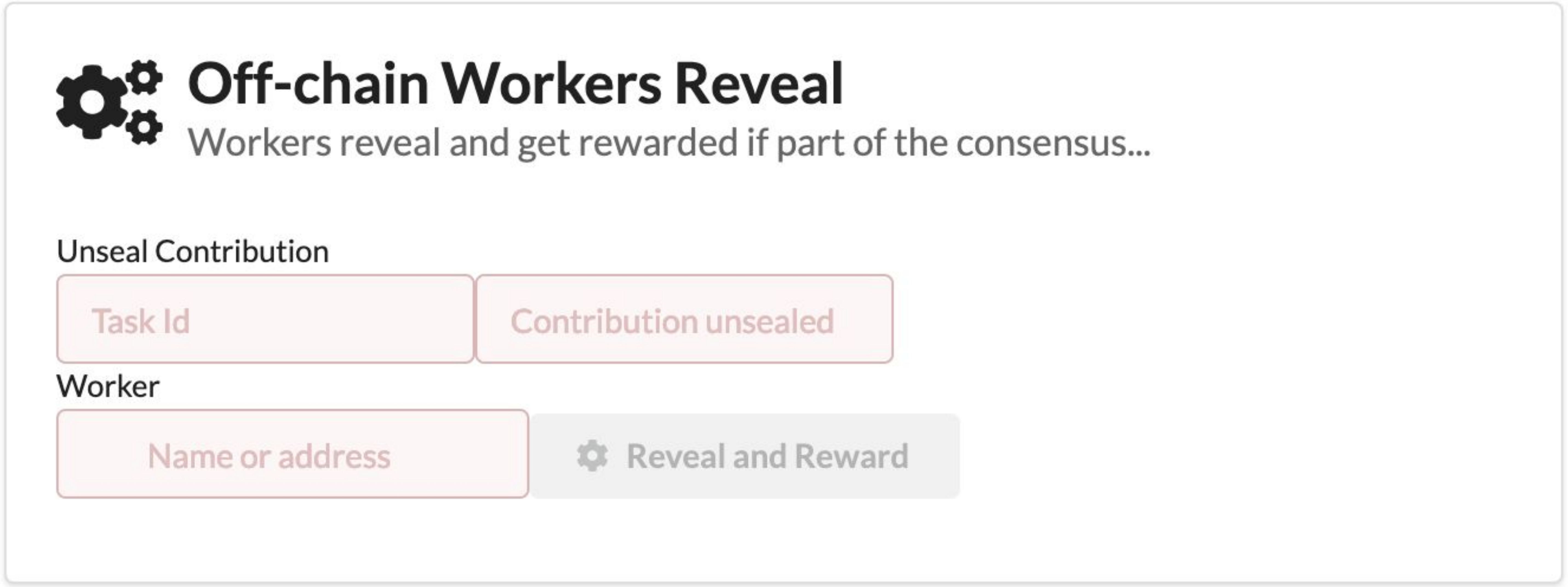
Workers stake for contribute ...

Contribution

Task Id | Contribution Vote | Contribution Sealed

Worker

Name or address | Stake And Contribute



Off-chain Workers Reveal

Workers reveal and get rewarded if part of the consensus...

Unseal Contribution

Task Id | Contribution unsealed

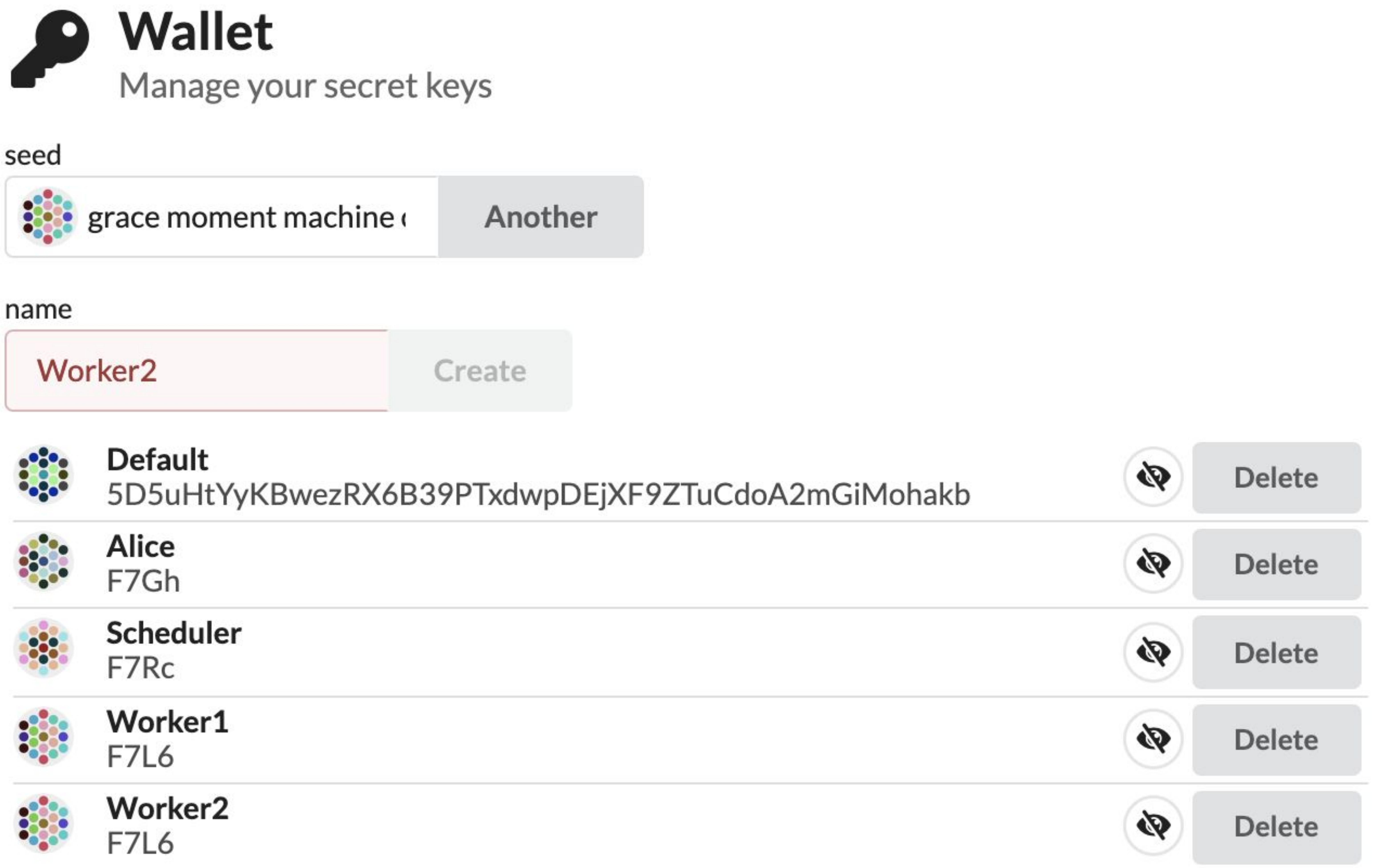
Worker

Name or address | Reveal and Reward

iExec

Actors

Substrate poc



Wallet

Manage your secret keys

seed

grace moment machine (

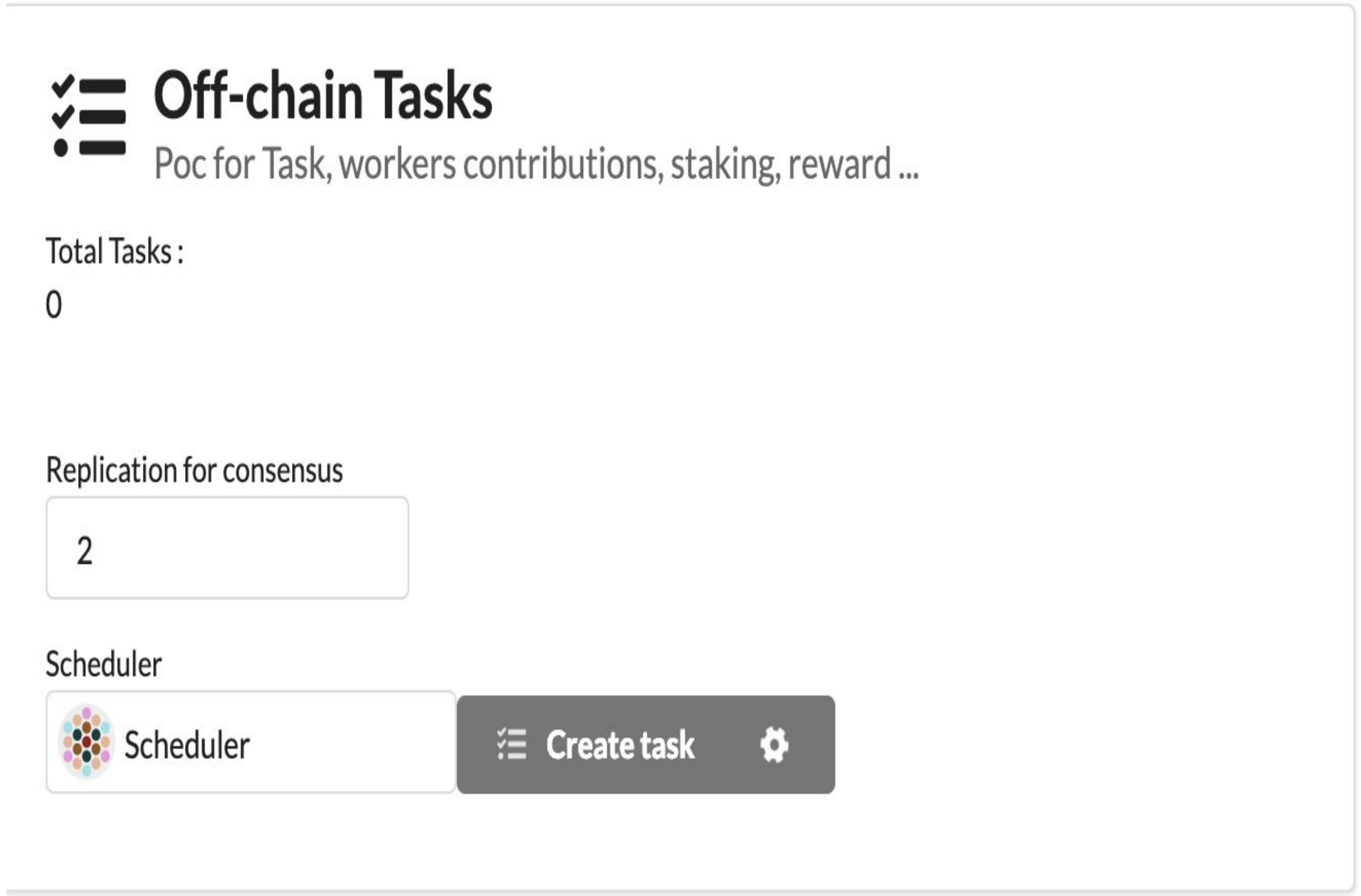
name

Worker2

Default 5D5uHtYyKBwezRX6B39PTxdwpDEjXF9ZTuCdoA2mGiMohakb		Delete
Alice F7Gh		Delete
Scheduler F7Rc		Delete
Worker1 F7L6		Delete
Worker2 F7L6		Delete

Task creation

Substrate poc



Off-chain Tasks

Poc for Task, workers contributions, staking, reward ...

Total Tasks :

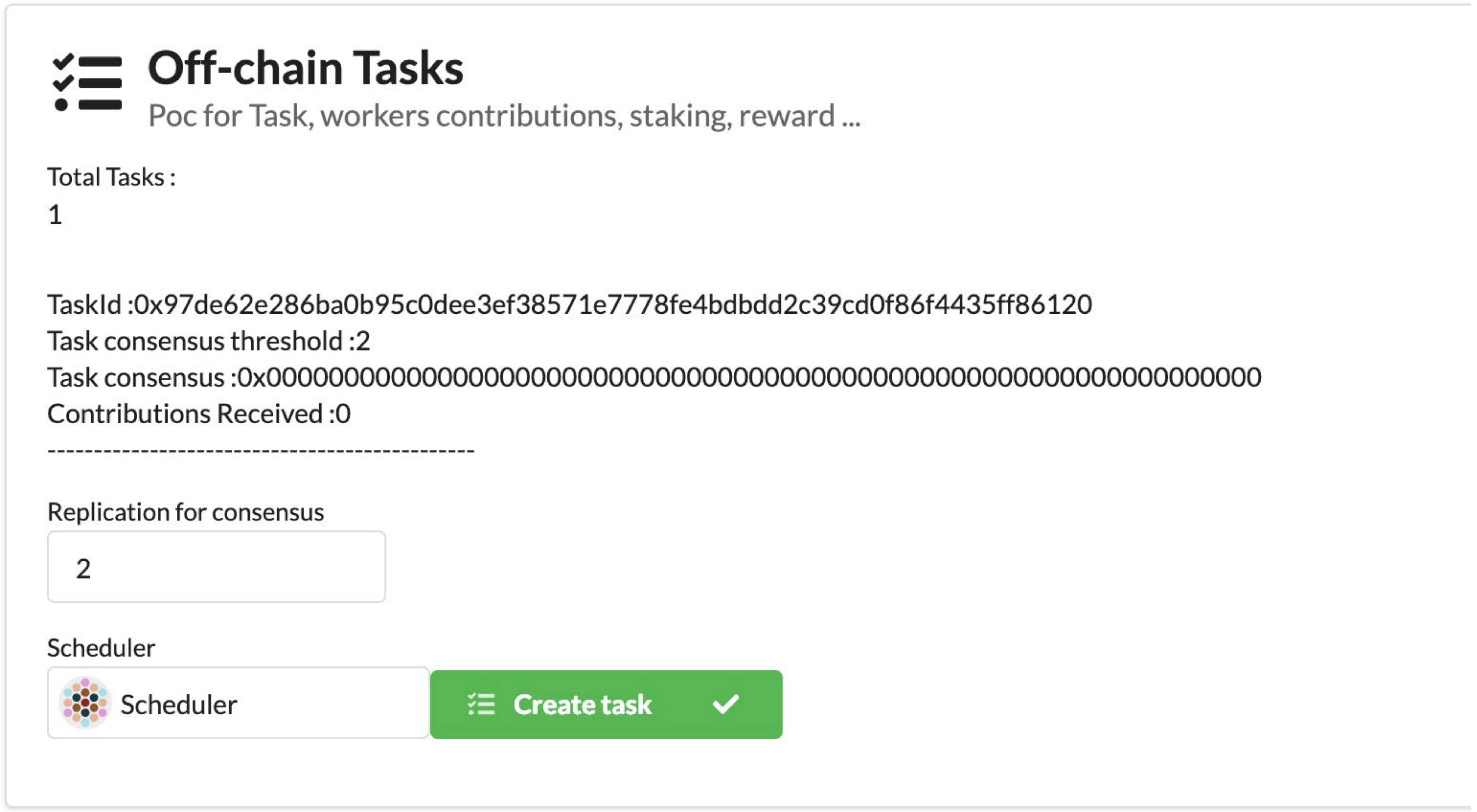
0

Replication for consensus

2

Scheduler

Scheduler



Off-chain Tasks

Poc for Task, workers contributions, staking, reward ...

Total Tasks :

1

```
TaskId :0x97de62e286ba0b95c0dee3ef38571e7778fe4bdbdd2c39cd0f86f4435ff86120
Task consensus threshold :2
Task consensus :0x0000000000000000000000000000000000000000000000000000000000000000
Contributions Received :0
----------------------------------------------
```

Replication for consensus

2

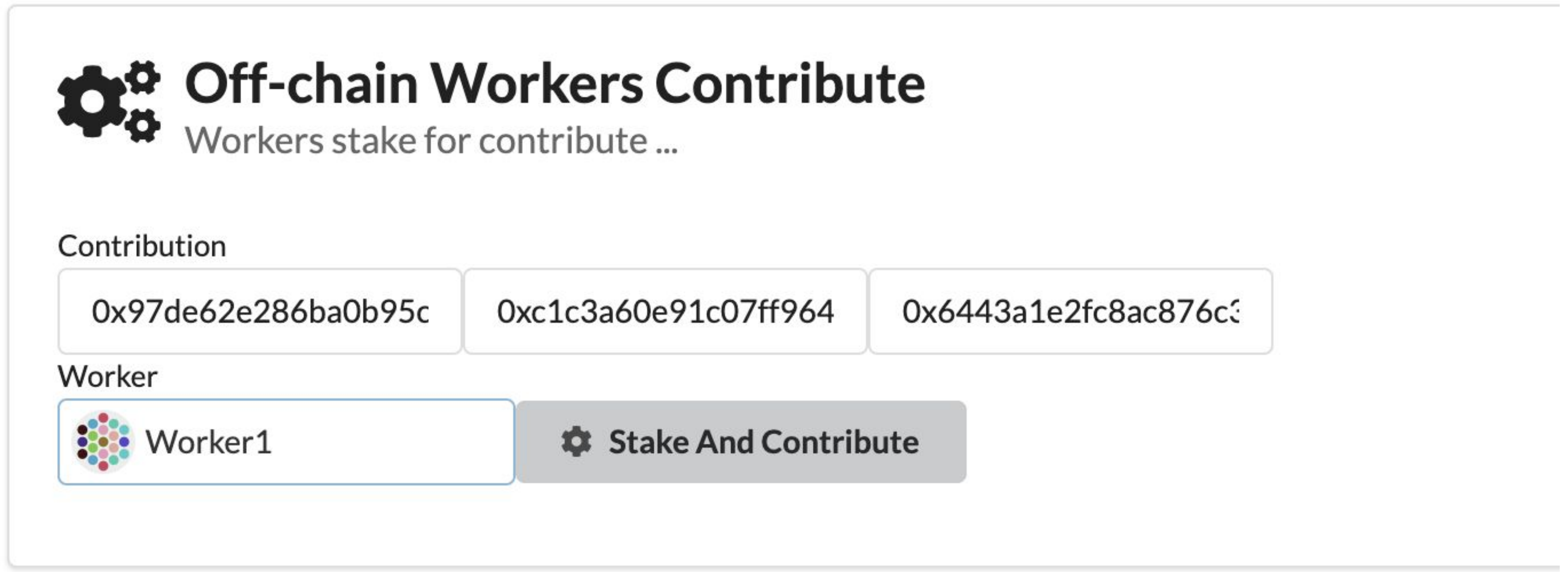
Scheduler

Scheduler

iExec

Worker contribute (stake)

Substrate poc



Off-chain Workers Contribute

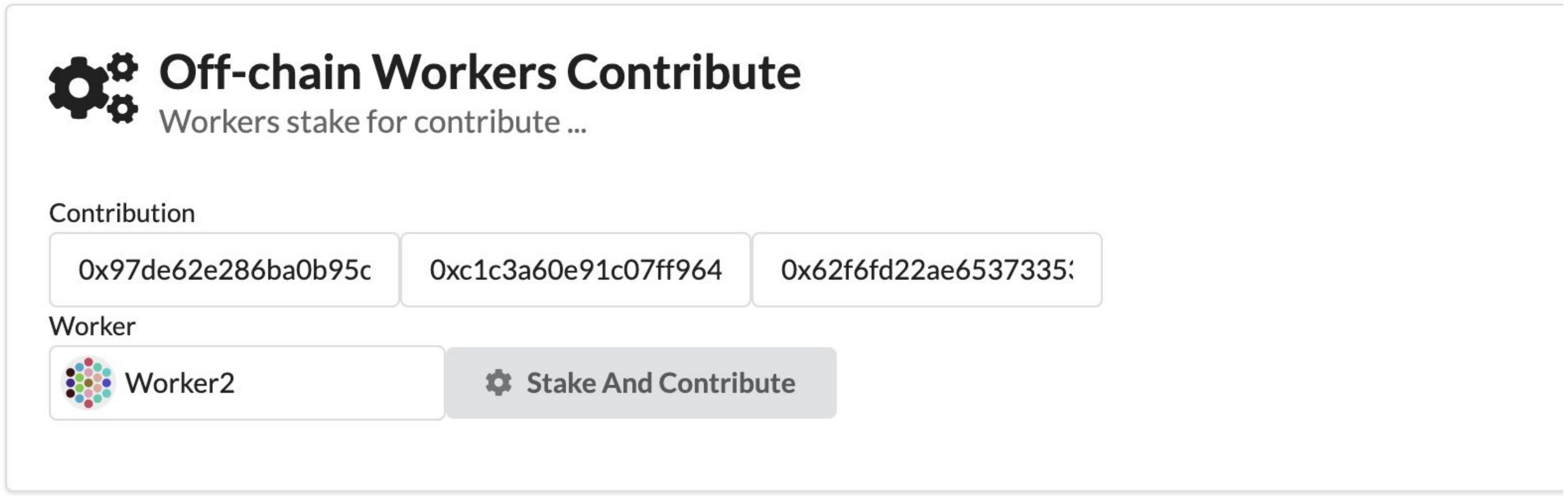
Workers stake for contribute ...

Contribution

0x97de62e286ba0b95c | 0xc1c3a60e91c07ff964 | 0x6443a1e2fc8ac876c3

Worker

Worker1



Off-chain Workers Contribute

Workers stake for contribute ...

Contribution

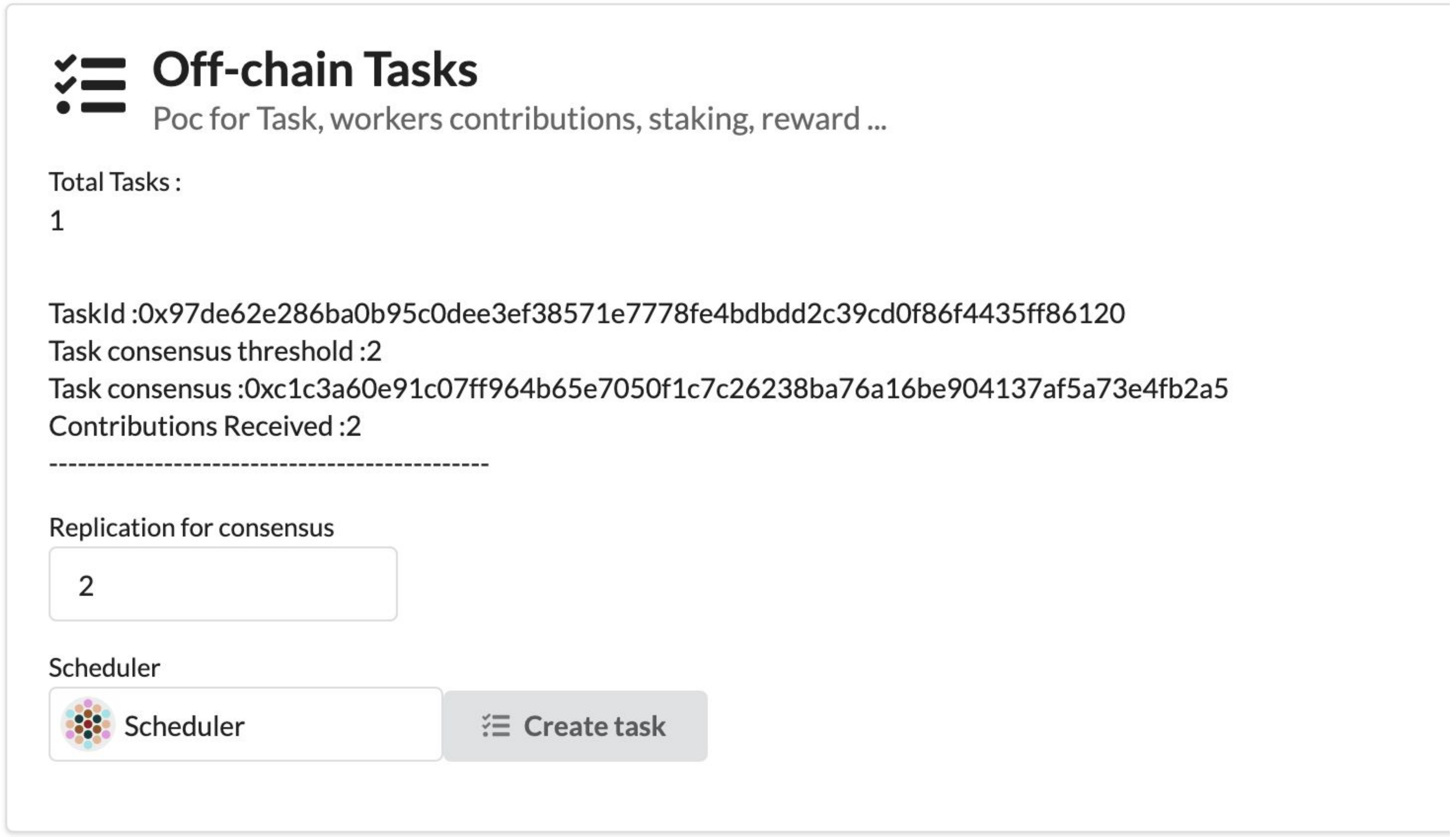
0x97de62e286ba0b95c | 0xc1c3a60e91c07ff964 | 0x62f6fd22ae6537335

Worker

Worker2

Consensus Reached

Substrate poc



Off-chain Tasks

Poc for Task, workers contributions, staking, reward ...

Total Tasks :
1

```
TaskId :0x97de62e286ba0b95c0dee3ef38571e7778fe4bdbdd2c39cd0f86f4435ff86120
Task consensus threshold :2
Task consensus :0xc1c3a60e91c07ff964b65e7050f1c7c26238ba76a16be904137af5a73e4fb2a5
Contributions Received :2
----------------------------------------------
```

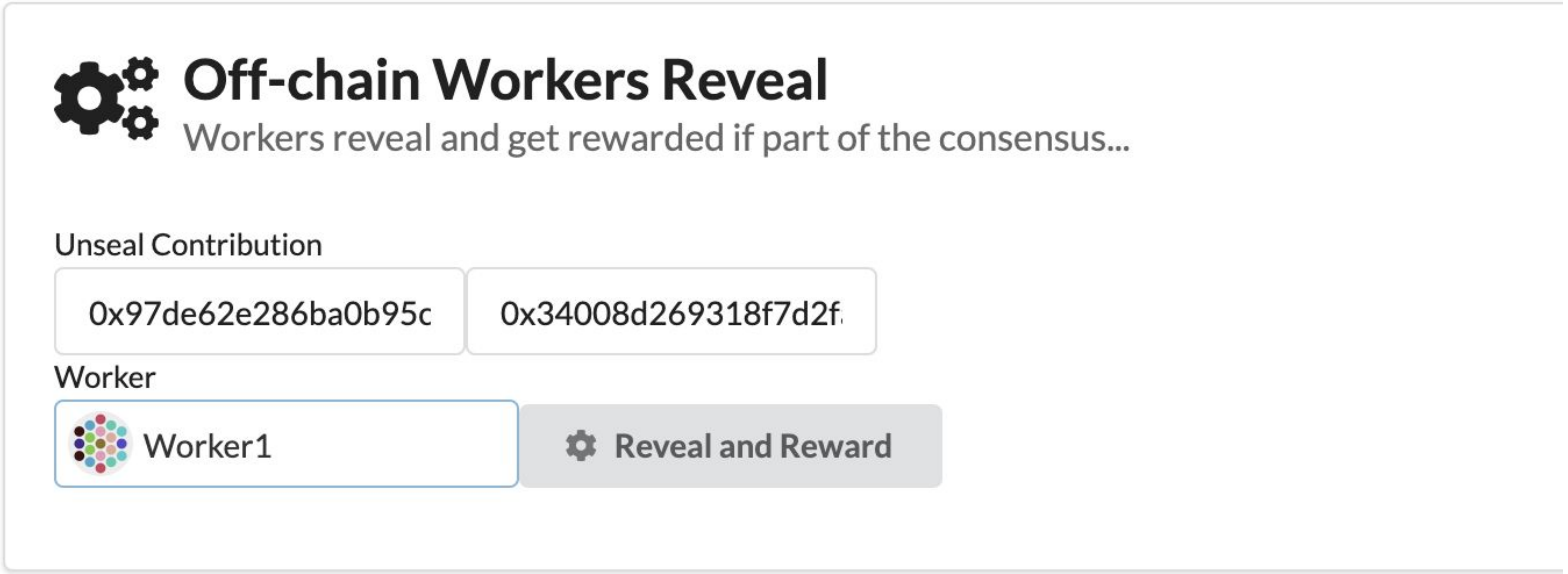
Replication for consensus: 2

Scheduler: Scheduler

iExec

Worker Reveal (reward or slashing)

Substrate poc



Off-chain Workers Reveal

Workers reveal and get rewarded if part of the consensus...

Unseal Contribution

0x97de62e286ba0b95c

0x34008d269318f7d2f

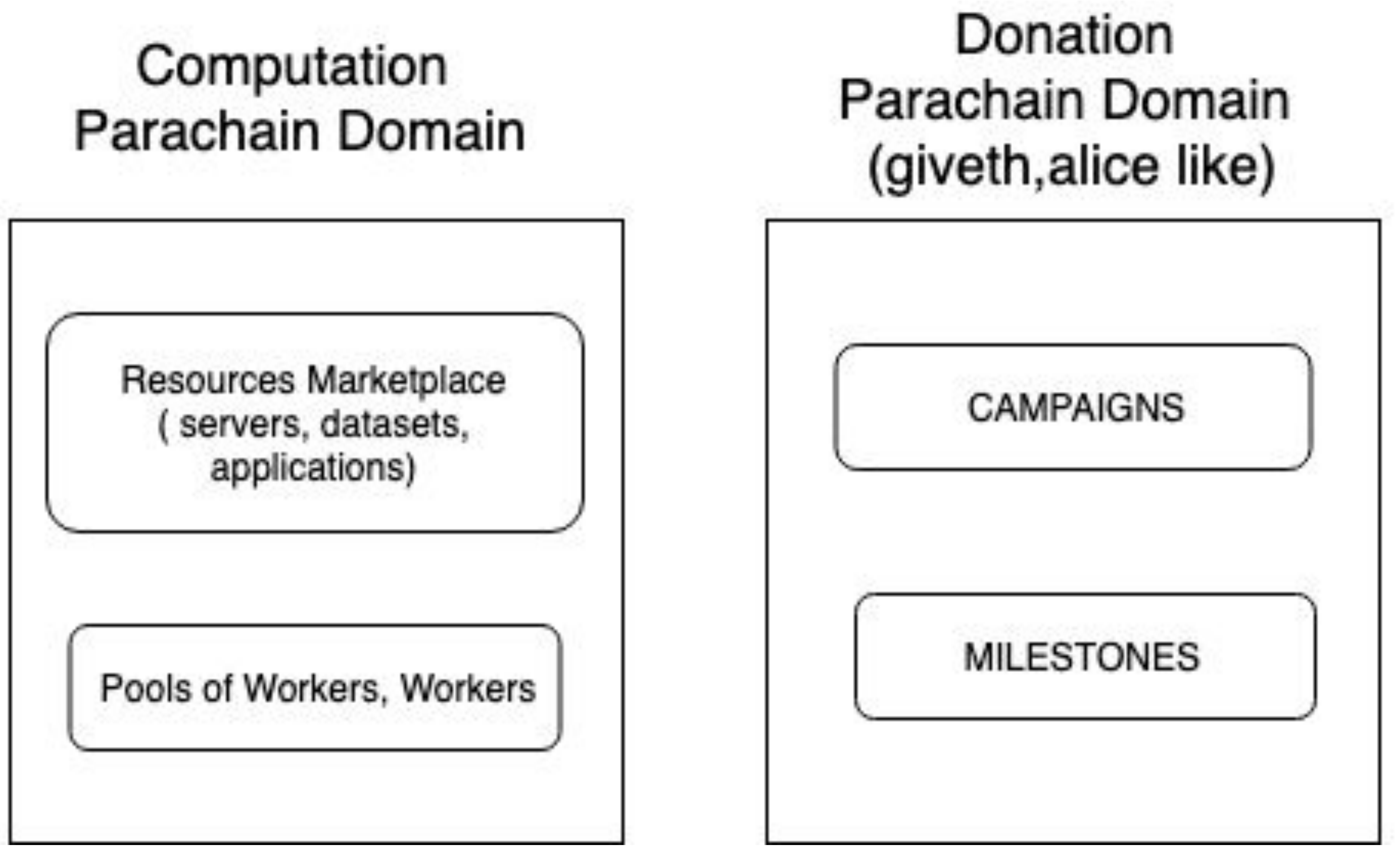
Worker

Worker1

Domains chains and interoperability

Speculative idea

- **Different parachain disparity typology (logic, consensus, security, governance, performances)**


Computation Parachain Domain
Resources Marketplace (servers, datasets, applications)
Pools of Workers, Workers
Donation Parachain Domain (giveth,alice like)
CAMPAIGNS
MILESTONES

Domain Constraints
Performance First,
Auditability, reputation,
Optimized for Micro payment
High throughput needed

Domain Constraints
Security First.
Security of money raised
Optimized for Large payment
Governance Model
to validate Milestones
Slow and safety ok

iExec

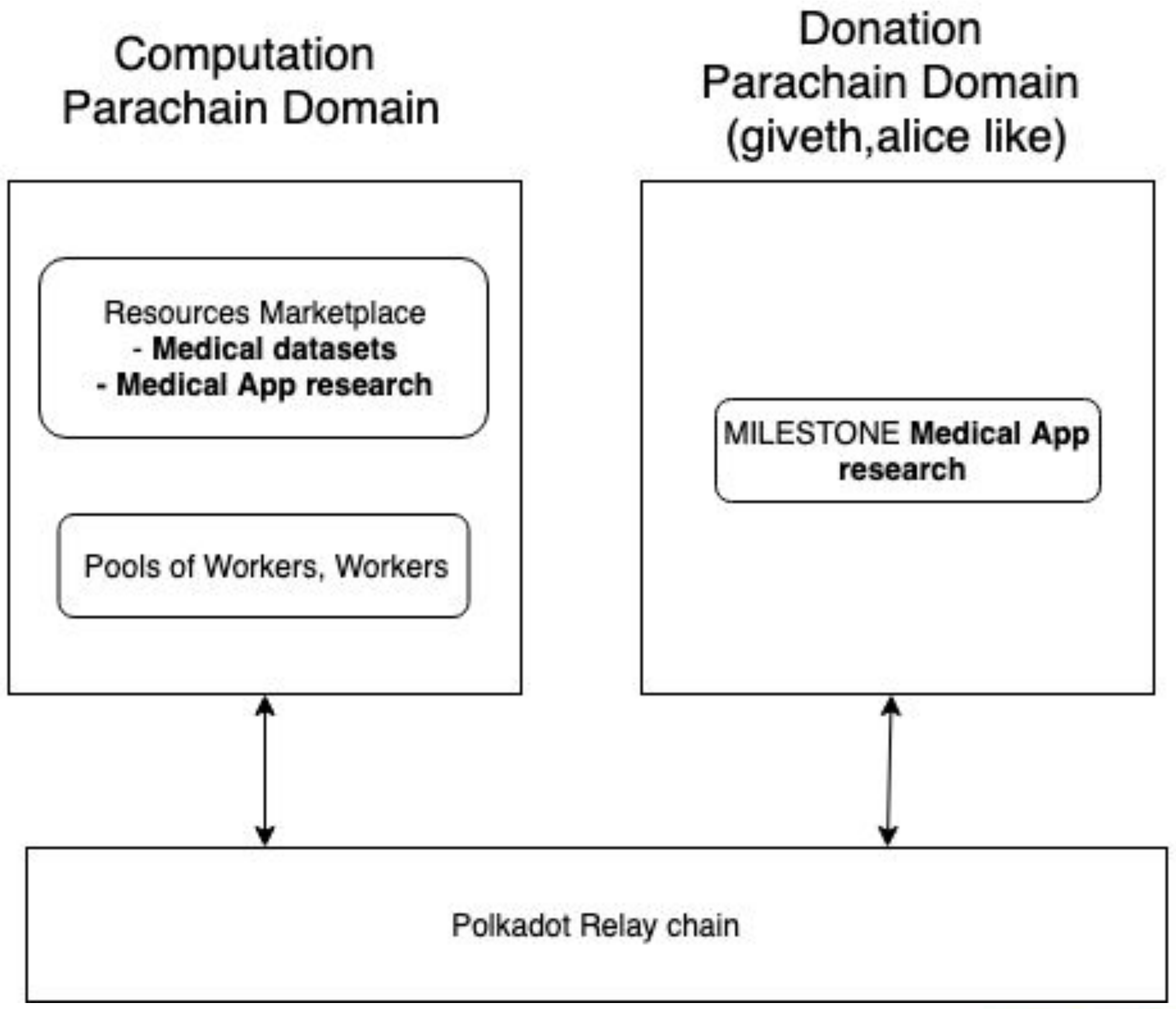
Domains chains and interoperability

Speculative idea

- Interoperability enable new use case :
  Trust lines between domains

- Donation + Computation :
  DCO : Donate computation offering ?

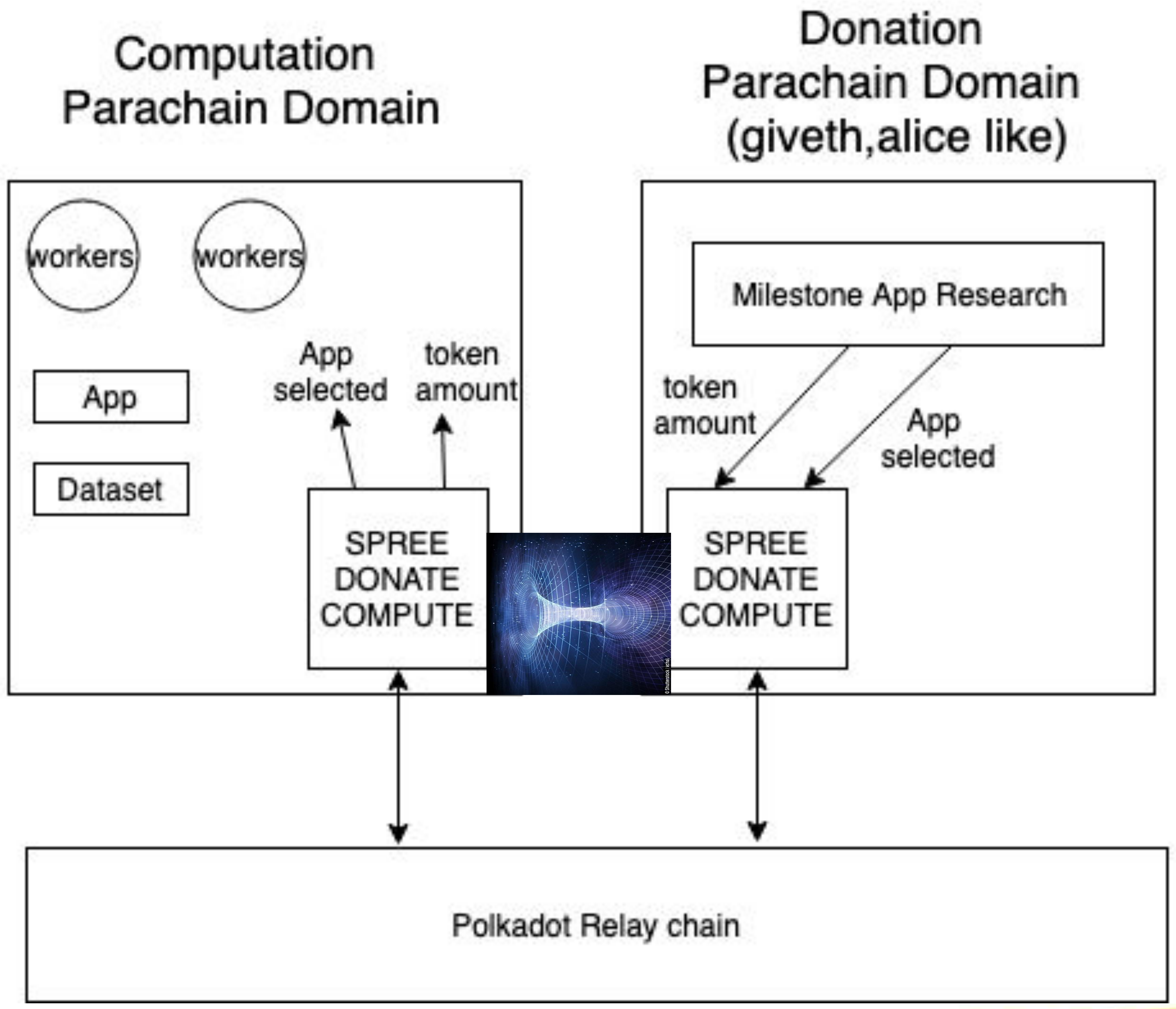
- Programmable money :
  Auditable proof that
  fund raised is well used
  (app medical compute research)


Computation Parachain Domain
Donation Parachain Domain (giveth,alice like)
Resources Marketplace - Medical datasets - Medical App research
Pools of Workers, Workers
MILESTONE Medical App research
Polkadot Relay chain

Domains chains and interoperability

Speculative idea

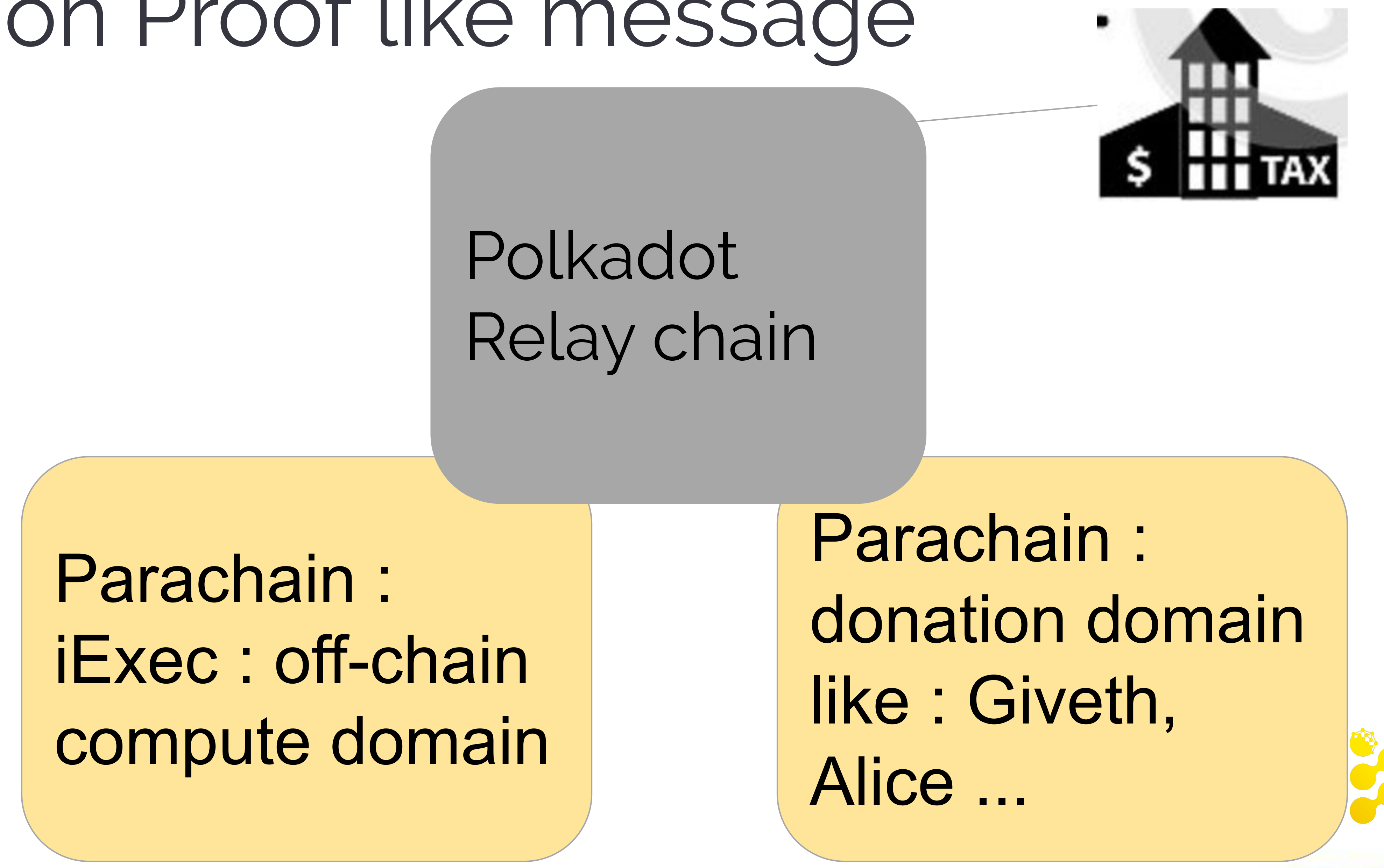
- DCO, Donate computation offering :code on **SPREE** ?
- SPREE (aka Trust wormholes) :
  Shared Protected Runtime Execution Enclaves
- Execution in a context of parachain but
  code definition as common shared lib
- Global integrity of monetary supply chosen
  and application selected



Domains chains and interoperability

Speculative idea

- But interoperability can for more than 2 chains
- Propagate proofs for third legal or government services
- Passing Some IDEN3 non-reusable donation Proof like message in the relay chain for tax service and obtain tax deduction.
- Ok to pay the relay chain message fees, if it reduce my bill taxe


$ TAX
Polkadot Relay chain
Parachain : iExec : off-chain compute domain
Parachain : donation domain like : Giveth, Alice ...

iExec

iExec

Thank you !

Francois Branciard
fb@iex.ec
@fbranciard
www.iex.ec