



# iExec

iExec - EthCC 2019

## iExec ecosystem in Web3 perspective

Francois Braniard  
[fb@iex.ec](mailto:fb@iex.ec)  
[@fbraniard](https://twitter.com/fbraniard)  
[www.iex.ec](http://www.iex.ec)

# Introduction

## **Substrate meets iExec: how to Build a Domain-Specific Token-Based Chain.**

Abstract:

The iExec network needs a Domain-Specific chain to track deals between the different actors interacting on its marketplace for cloud resources (buyers and sellers of computing, datasets and applications). The chain would be used to record all contributions of these actors to trigger payments using PoCo (The iExec Proof-of-Contributon algorithm).

For private and consortium use of the iExec Stack, a PoA bridge and chain can be set up to offer an enterprise-ready solution.

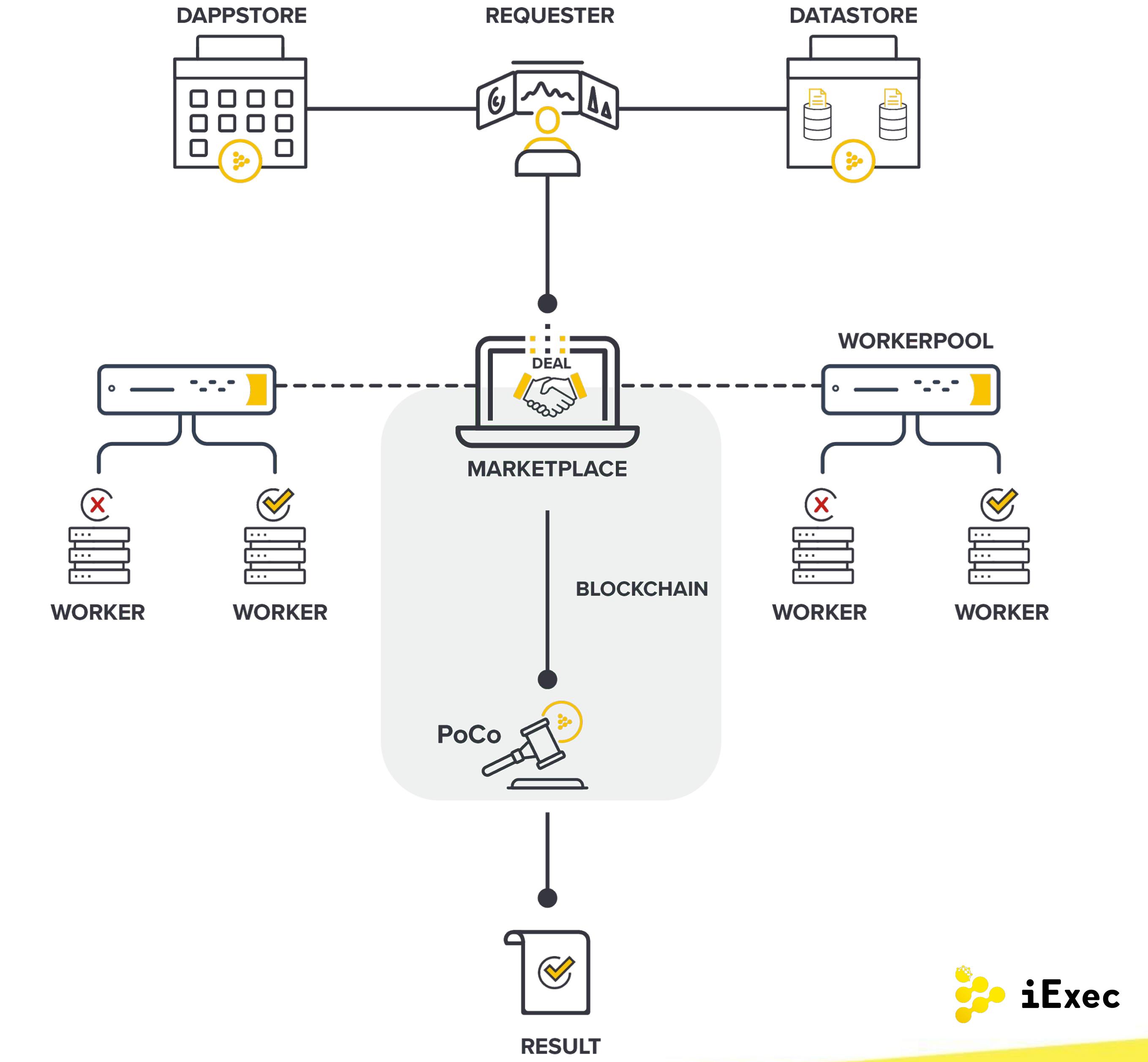
For public usage, iExec has more complex requirements and is experimenting with Substrate to tackle: a token-based PoS consensus, governance, upgradability, the use of wasm for new features, and interoperability as a Polkadot parachain.



# Plan

- 1. iExec ecosystem**
- 2. iExec ecosystem in a web3 perspective**
- 3. Delivery in progress : Consortium bridge**
- 4. Research in progress : Substrate Domain-Specific Token-Base Chain**

# iExec ecosystem



# 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>

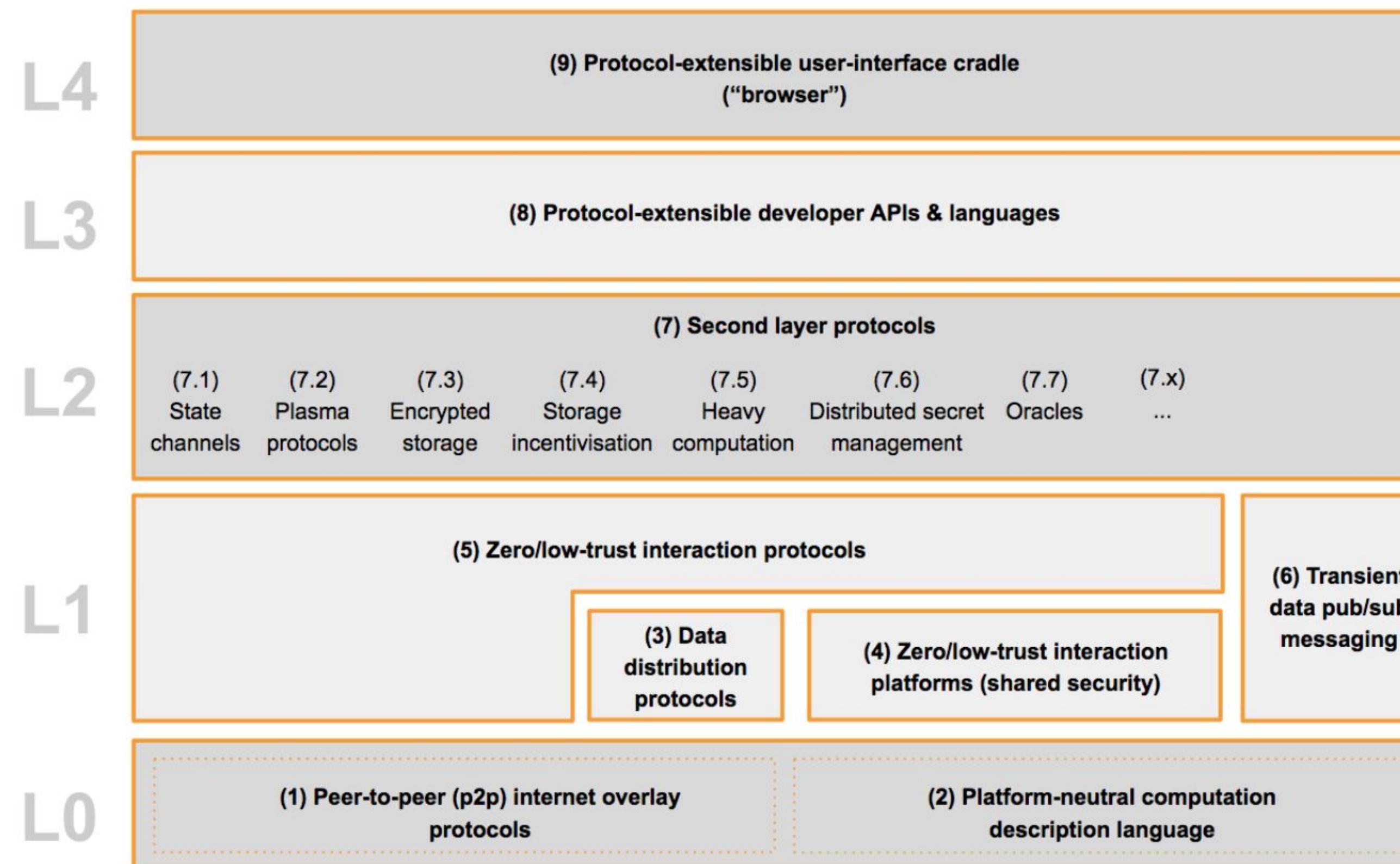


# Computation layer

computation layer for the decentralized web3 stack

iExec ecosystem in a web3 perspective

## Web3 Tech Stack



(7) Second layer protocols

(7.5) Heavy computation



# Sidechain strategy

iExec ecosystem in a web3 perspective

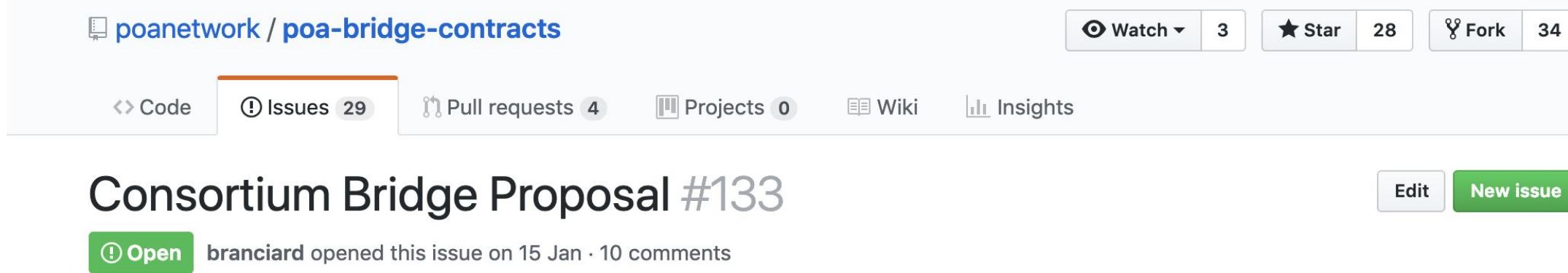
- **Deliver** pragmatic intermediary solution with existing technologies :
  - Mainnet EVM smart contract
  - PoA chain EVM smart contract
  - Bridges (EVM <-> EVM)
  - Mitigate governance with consortium
- **Research** on an optimal solution :
  - Replace bridges by dedicated relay-chain, with their 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



- <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

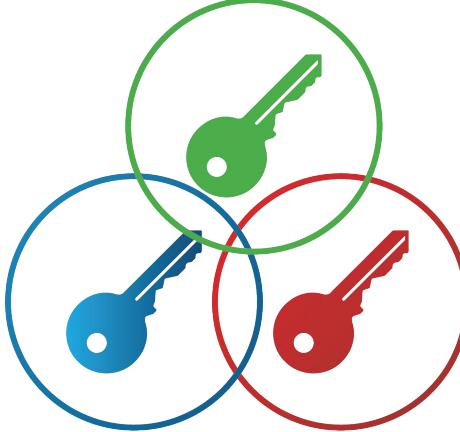
Consortium bridge

## 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**

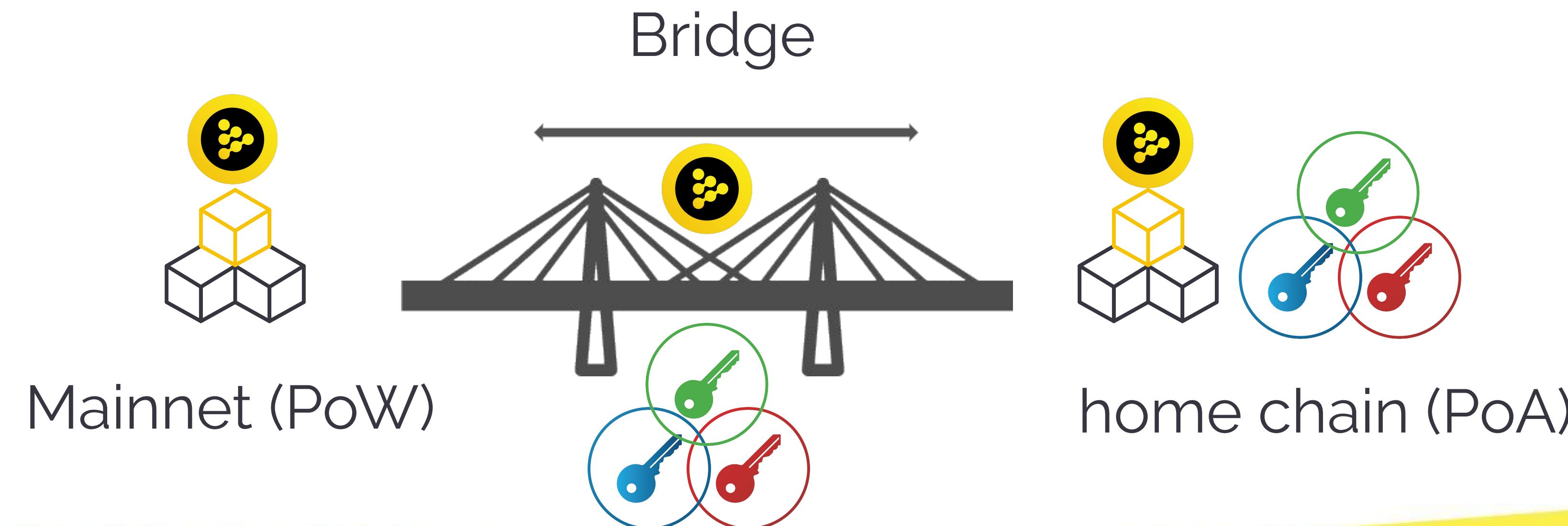


**iExec**

# Let's automate token payment between them

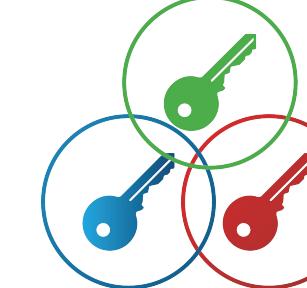
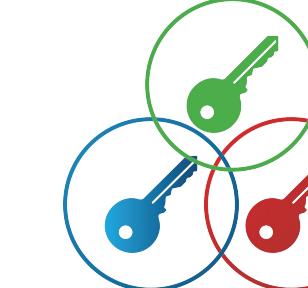
Consortium bridge

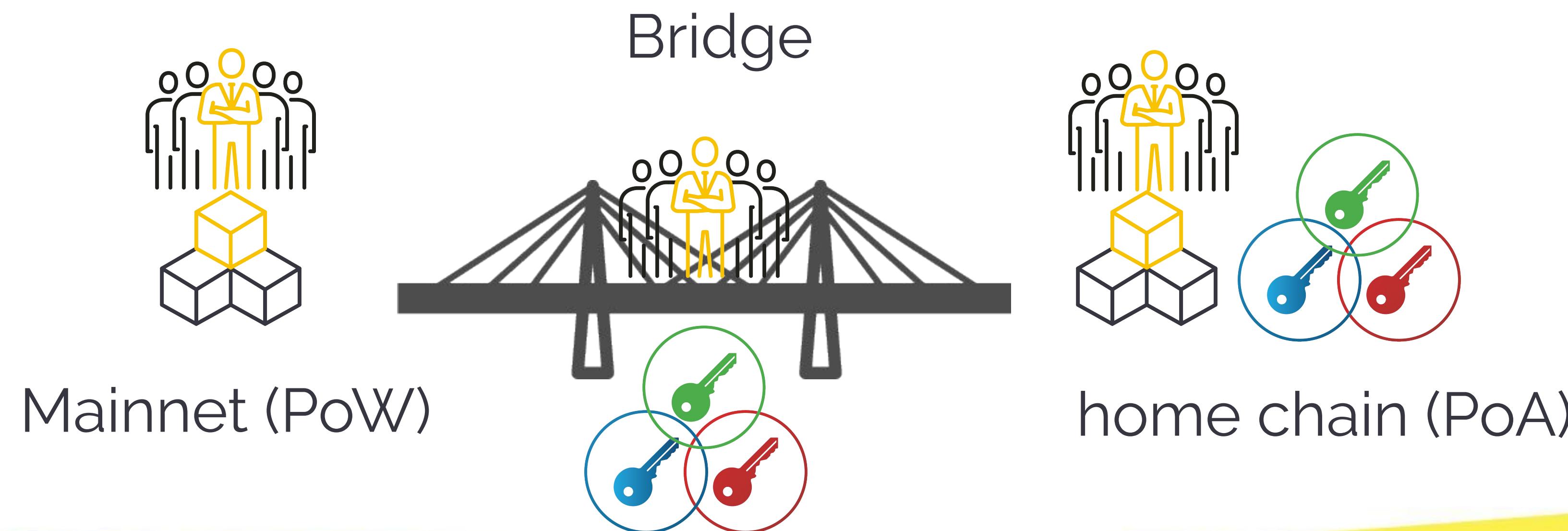
- add a **consortium bridge** under same authority than can **deposit** and **withdraw** between the mainnet and their “home” chain.
- use marketplace contract to set dataset, app, resources dynamic **prices usage between them**.



# Let's automate payment for their join customers too

Consortium bridge

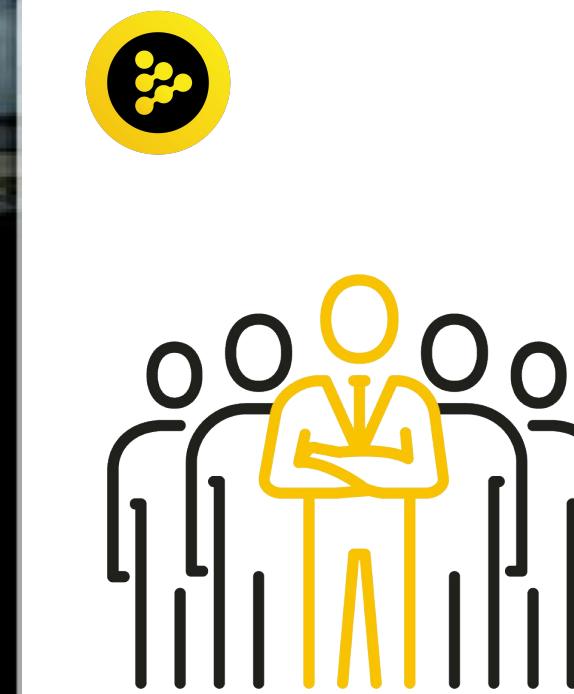
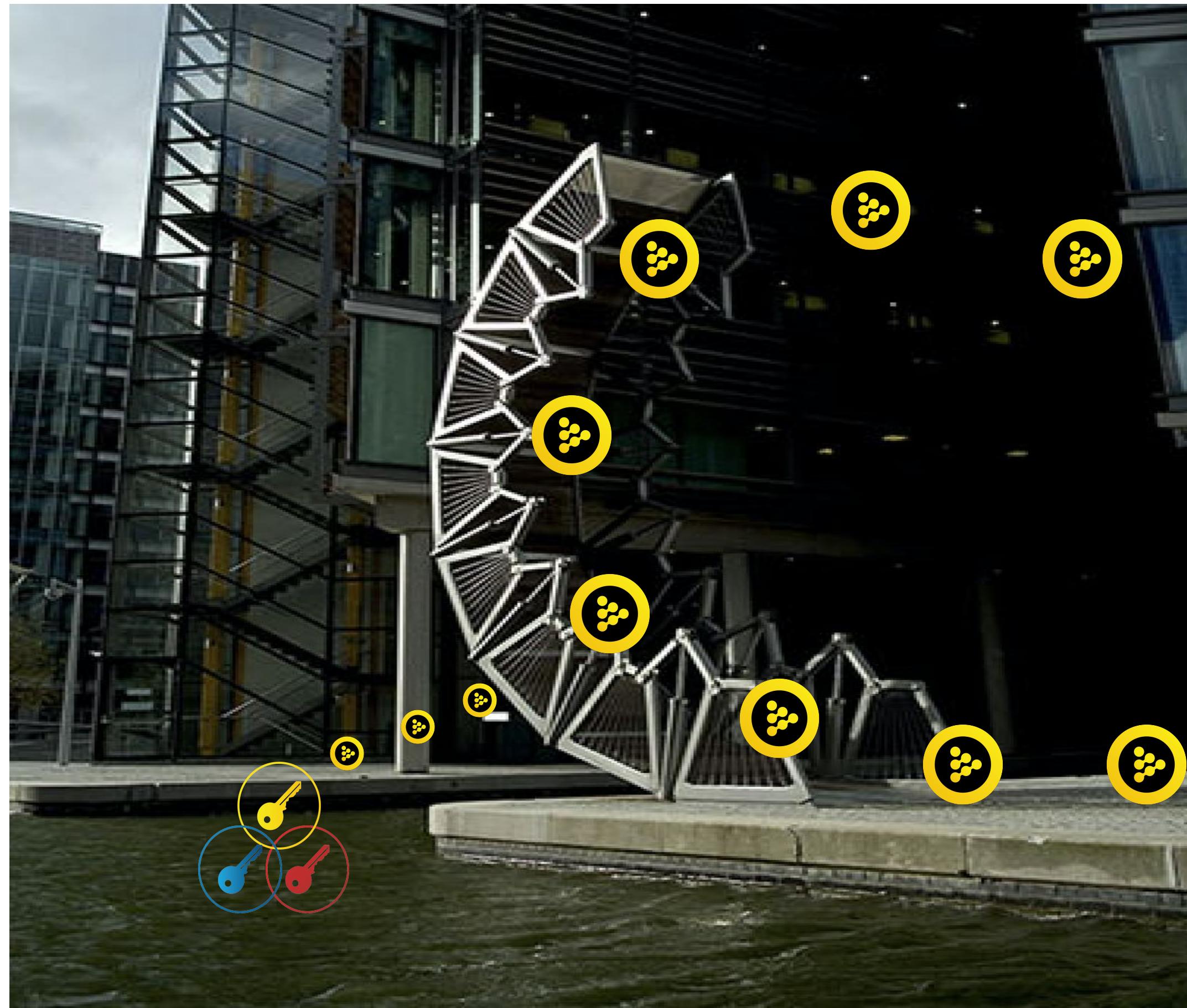
-  a potential customer of
- IF (  trust  ) AND (  WHITELIST  )



# What happens if you are not whitelisted to cross consortium bridge ?

Consortium bridge

consortium side



Mainnet side

asset return back  
(minus bridge fees)



# Research in progress : Substrate POC

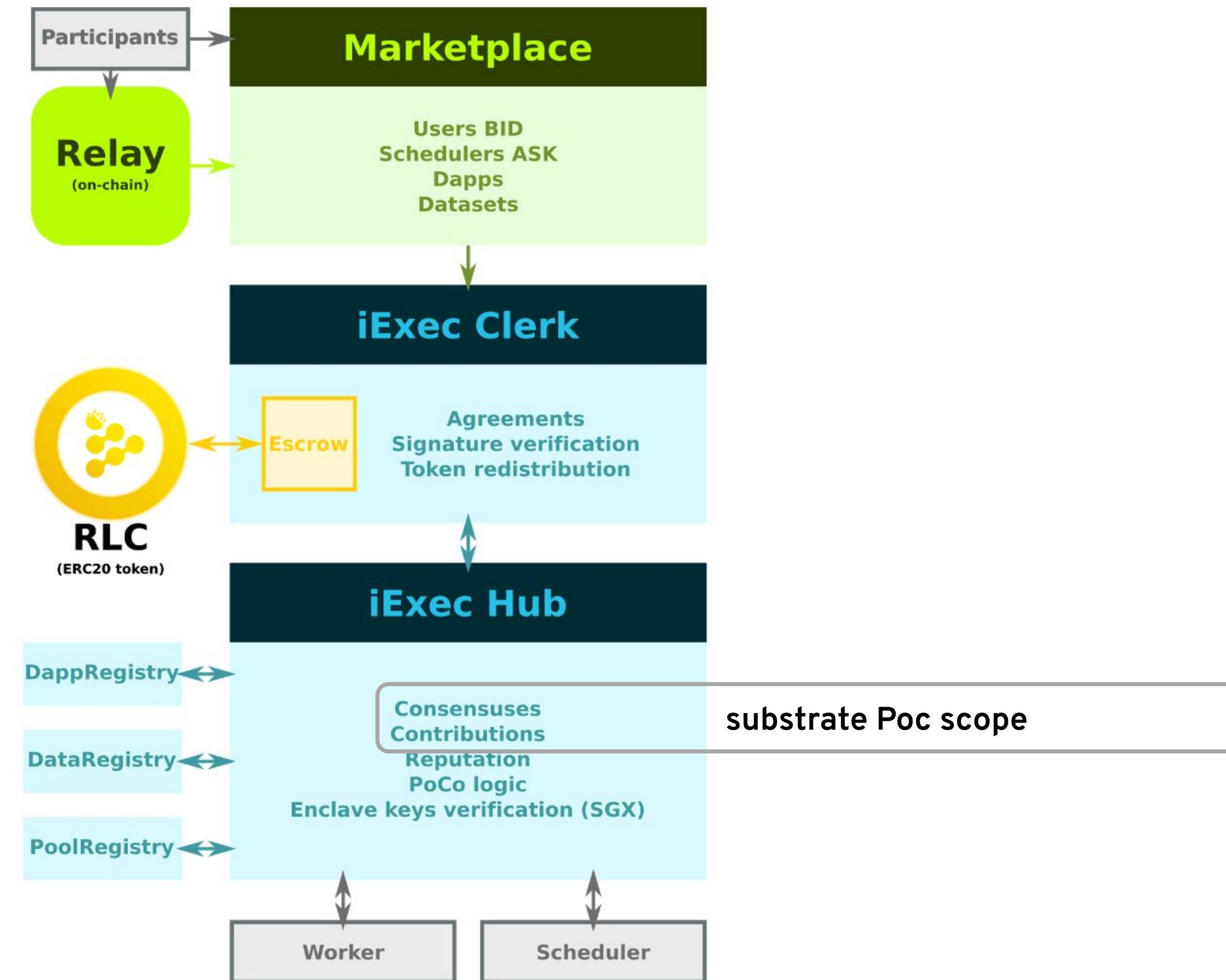
## Substrate POC

- 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



# Contribute / Reveal

Substrate POC

1. Off chain Task : echo “hello-world”
2. contribute :
  - **stake to contribute**
  - contribution 2 params :
    - result vote : vote on a result but do not reveal it yet.
    - contribution seal : needed to prove result knowledge later
3. consensus reached on vote
4. reveal
  - **reveal** and **reward/slash**
  - reveal 1 param :
    - contribution unseal

# Contribute / Reveal

Substrate POC

1. Off chain Task : echo “hello-world”
2. contribute :
  - **stake** to **contribute**
  - contribution 2 params :
    - **result vote** : hash(hash(hello-world))
    - **contribution seal** : hash(@worker ^ hash(hello-world))
3. consensus reached on vote
4. reveal
  - **reveal** and **reward/slash**
  - reveal 1 param :
    - **contribution unseal** : hash(hello-world)
      - check result vote was valid
      - contribution seal was valid.

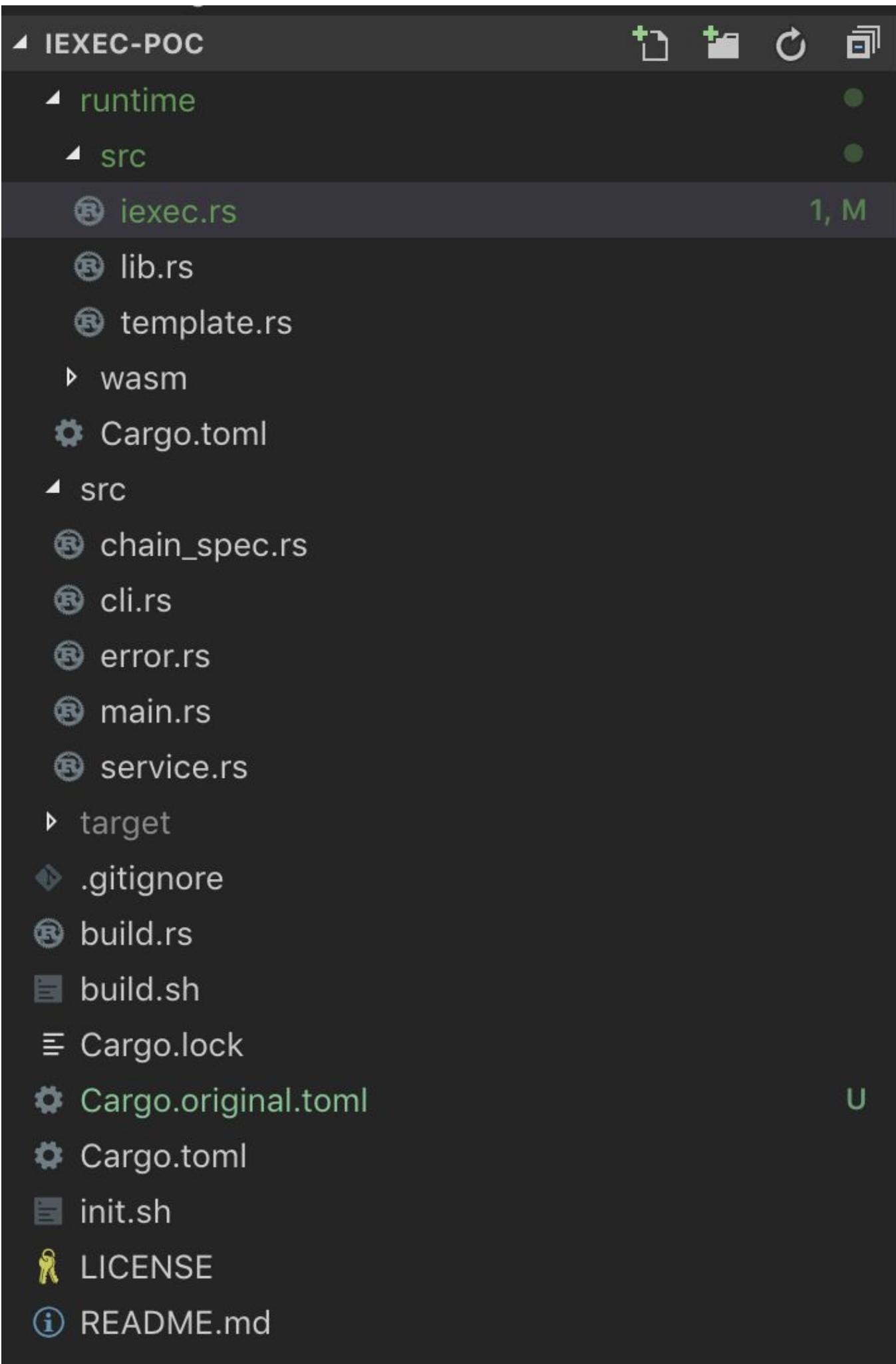
# Demo

## Substrate POC



# From substrate node template

Substrate poc



```
construct_runtime!{
    pub enum Runtime 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\_trustmanagement.pdf
}

#[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 {
```

# Substrate-ui

The screenshot shows a web browser window with the URL `localhost:8000`. The main content area displays two sections: "Wallet" and "Off-chain Tasks".

**Wallet Section:**

- Seed:** A text input field containing "Some seed for this key".
- Another:** A button.
- Name:** A text input field containing "A name for this key".
- Create:** A button.
- Default:** A row with a circular icon, the name "Default", and a public key "5D5uHtYyKBwezRX6B39PTxdwpDEjXF9ZTuCdoA2mGiMohakb". It includes a "Delete" button.
- Alice:** A row with a circular icon, the name "Alice", and a public key "F7Gh". It includes a "Delete" button.

**Off-chain Tasks Section:**

- Total Tasks:** Displays "0".
- Replication for consensus:** A text input field containing "Replication for consensi".
- Scheduler:** A text input field containing "Name or address".
- Create task:** A button.

**Developer Tools Console:**

- Messages:** Shows 3 messages, 3 user messages, 1 warning, and 2 info.
- Expression:** Shows the message "not available".
- Warning:** An alert about MetaMask privacy mode.
- Code Snippets:** Shows code snippets for `initialiseFromMetadata`, `runtime.iexec.allTasksArray`, and `f`.
- Object Properties:** A detailed list of properties for the `allTasksArray` object, including `allTasksArray`, `allTasksCount`, `allTasksIndex`, `contributions`, `contributionsArray`, `contributionsCount`, `contributionsIndex`, `contributionsResultVoteCount`, `moduleSalt`, `something`, `tasks`, `tasksConsensus`, `constructor`, `hasOwnProperty`, `isPrototypeOf`, `propertyIsEnumerable`, `toLocaleString`, `toString`, `valueOf`, and `__defineGetter__`.

# Contribution reveal

Substrate poc



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



# Actors

Substrate poc

 **Wallet**  
Manage your secret keys

seed

 grace moment machine < [Another](#)

name

 **Worker2** [Create](#)

	<b>Default</b> 5D5uHtYyKBwezRX6B39PTxdwpDEjXF9ZTuCdoA2mGiMohakb	 <a href="#">Delete</a>
	<b>Alice</b> F7Gh	 <a href="#">Delete</a>
	<b>Scheduler</b> F7Rc	 <a href="#">Delete</a>
	<b>Worker1</b> F7L6	 <a href="#">Delete</a>
	<b>Worker2</b> F7L6	 <a href="#">Delete</a>

# Task creation

# Substrate poc

The screenshot shows a user interface for managing off-chain tasks. At the top left is a logo consisting of three horizontal bars with checkmarks. To its right is the title "Off-chain Tasks" in large, bold, black font. Below the title is a subtitle "Poc for Task, workers contributions, staking, reward ...". On the left side, there's a section labeled "Total Tasks :" followed by the number "0". In the center, there's a section labeled "Replication for consensus" with the number "2" displayed inside a light gray rounded rectangle. At the bottom, there's a "Scheduler" section containing a circular icon with colored dots and the word "Scheduler". To the right of this is a dark gray button with the text "Create task" and a gear icon.

# Worker contribute (stake)

Substrate poc

 **Off-chain Workers Contribute**  
Workers stake for contribute ...

Contribution

0x97de62e286ba0b95c    0xc1c3a60e91c07ff964    0x6443a1e2fc8ac876c3

Worker

 Worker1     Stake And Contribute

 **Off-chain Workers Contribute**  
Workers stake for contribute ...

Contribution

0x97de62e286ba0b95c    0xc1c3a60e91c07ff964    0x62f6fd22ae6537335:

Worker

 Worker2     Stake And Contribute

# 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  Create task



# 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

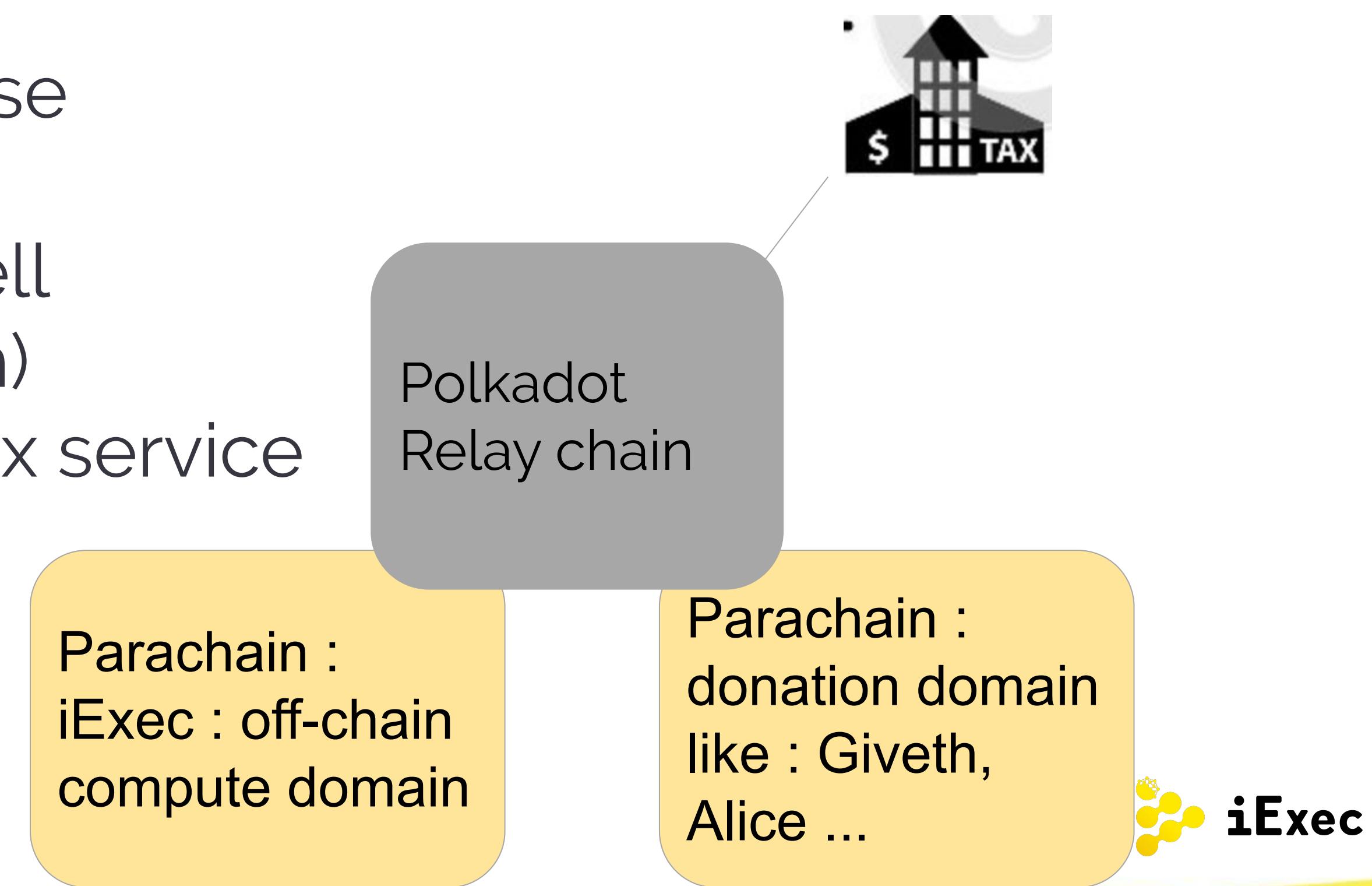


Reveal and Reward

# Speculative idea with interoperability

Substrate poc

- Different parachain typology (logic, consensus, security, governance, performances )
- But interoperability enable new use case
- DCO : Donate computation offering ?
  - Auditable proof that fund raised is well used (app medical compute research)
  - External proof in the relay chain for tax service to have tax deduction.  
(IDEN3 non-reusable proof like : ) )



# Common vision and synergy

Substrate meets iExec | iExec meets Substrate

## Substrate

Off-chain compute

Allows for off-loading jobs outside of the runtime

- Chain can schedule tasks to be executed by a recognised group
- Task definition stored as Wasm on-chain
- Additional APIs for this Wasm... URL fetch, local storage, IPFS
- Create economic games on top to ensure fidelity of result



**POCO**



Credit and extract from Gavin Wood talk at M-1 | Asset Management 3.0 Conference | Day 2  
[https://www.youtube.com/watch?v=Opj\\_5ORbyXA](https://www.youtube.com/watch?v=Opj_5ORbyXA)



# iExec

Thank you !

Francois Braniard  
[fb@iex.ec](mailto:fb@iex.ec)  
[@fbraniard](https://twitter.com/fbraniard)  
[www.iex.ec](http://www.iex.ec)