Open Validator Service Broker

This draft spec is based off of 1. Kubernetes Service Catalog's and Operators & 2.Open Service Broker API

Some basic knowledge of Kubernetes and Infrastructure as a Service is assumed.

Motivation

We are proposing that Lido Finance adopts a formal specification for implementation by its Node Operator's (those that are maintaining and operating the underlying infrastructure that Lido uses for its stakers) for a Service Broker to help facilitate Client Diversity and Node Operator interoperability across their infrastructure. By adopting such requirements, it will enable Lido Finance users (re: stakers) to request that their stake be used to allocate/provision a certain client, which can be informed by potential incentives by Lido Finance for encouraging or discouraging certain choices.

A service broker provides an interface between an service provider (e.g. GCP, Azure or AWS), and an application platform (e.g. Kubernetes or Cloud Foundry). The service broker is managed by platform operators (Node Operators in Lido's case).

These platform/node operators are responsible for configuring the broker to meet the needs of their network, platform(w.r.t. Staking requirements, e.g. uptime, availability, slashing mitigation, etc), and developers(including users). Developers could possibly use the broker to provision and bind new services to their applications.

An example of this would be a team wanting to provide RPC access to its managed node's for their own users (this could end up being like a 'Infura' like service where a team uses their staked ETH and can access the equivalent nodes that their stake would provision).

Therefore, a service broker is responsible for federating access between an application provider and a developer with respecting the wishes of the platform and its operators. Each of these parties influences the broker, its services, and structure.

Application: Client Diversity

Client diversity is simply ensuring that not one client implementation is relied on to the extent that if an issue was to arise it would not cause a cascading or catastrophic failure. Using multiple different clients is especially important post-Merge to reduce the probability of a bug causing a halt of the chain, as the requirements for such a scenario are smaller than pre-Merge.

With a Node Operator having this capability, Lido Finance can incentivize stakers to request that their 'representative validator' should be running a specific client that is chosen from the service catalog

.

- Fetching the catalog of backing services that a service broker offers

The catalog describes all of the services that can be provisioned through a service broker, and each service is made up of a specific Client Implementation and version, Execution layer specifics, Engine API version, etc. A service can bundle multiple clients or just simply have one. It can even be middleware (e.g. MEV-Boost).

- Provisioning new service instances

A service instance is a provisioned instance of a service and plan as described in the service broker's catalog.

- Connecting and disconnecting applications and containers from those service instances
- Deprovisioning service instances

This API Specification is based off of what we use to model our infrastructure internally at Manifold Finance. You can see the draft specification here on GitHub, GitHub - manifoldfinance/open-validator-operator: Open Validator Operators - specification and design documents

Feedback

Feedback is always welcomed. This was written based off of only publicly available information that I could find w.r.t. Node Operator requirements from Lido.

Disclosure: I am a co-founder at Manifold Finance, and we also have a submitted application to become a Node Operator for Ethereum Mainnet for Lido Finance. I own a position in Lido Finance, Manifold Finance, Sushiswap, Yearn Finance, Rocket Pool, Rari Capital.