Edge-LB

Operationalizing Ingress Load Balancing

May 2018

Edge-LB Overview

A DC/OS cluster will have a variety of load balancing/proxy requirements dependent on the DC/OS architecture, deployed services, size, and several other factors

Edge-LB:

- Delivered with Enterprise DC/OS 1.10
- Ingress Load Balancer
- Based on HAProxy (pluggable in the future)

Looking Back: Marathon-LB

Marathon-lb is the predecessor to Edge-LB

Marathon-LB consists of a single container (based on a Docker image) that handles all processing - config generation, HAProxy reloads, and the HAProxy process itself

Marathon-lb listens to the Marathon event bus and therefore is limited to providing load balanced service to Marathon applications

Looking Forward: Edge-LB

Edge-LB is the successor to Marathon-LB

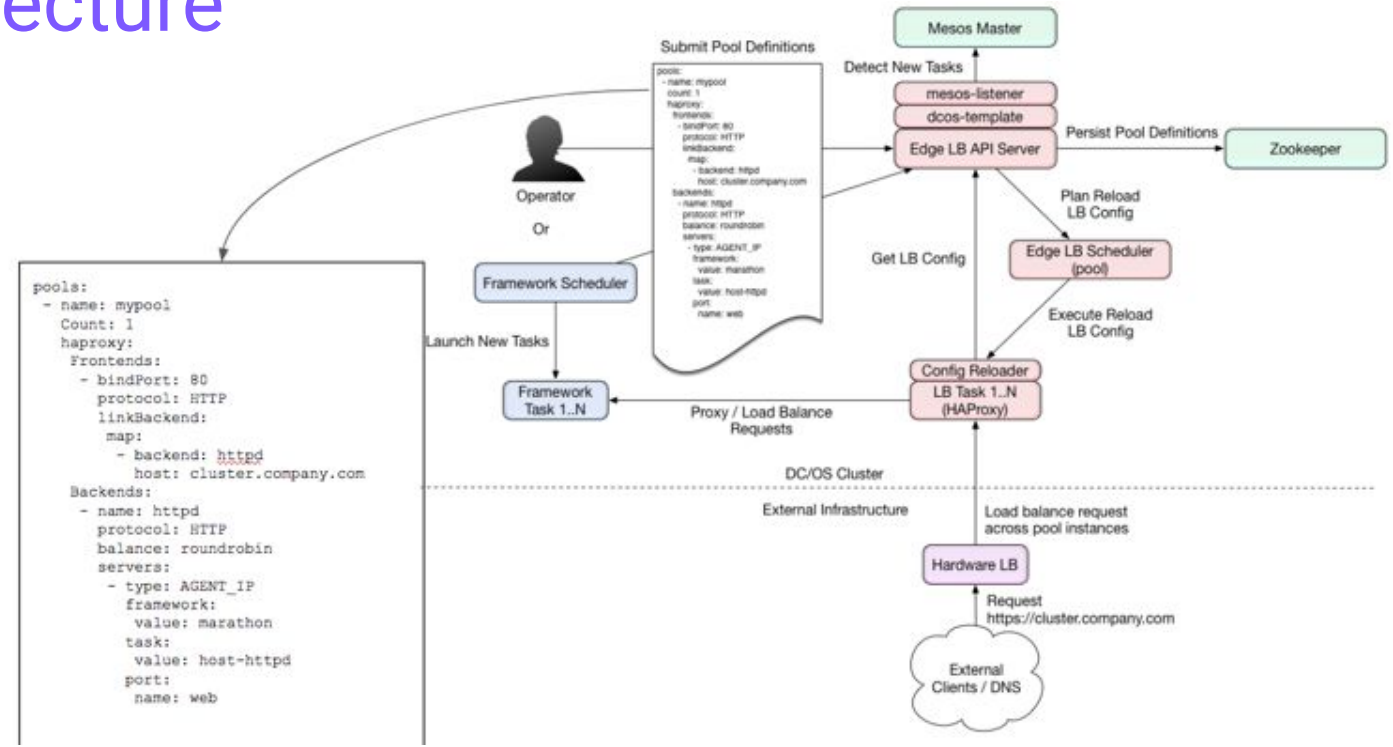
Edge-LB consists of a three-tier architecture

- Edge-LB API Server
  - Presents an API to submit configurations to and handles config generation
- Edge-LB Pool Server
  - dcos-commons -based framework scheduler that manages Edge-LB Load Balancers
- Edge-LB Load Balancer
  - Runs the HAProxy instances

Edge-LB is integrated with mesos and therefore has the ability to load balance any service running on DC/OS (not limited to Marathon)

Edge-LB also provides much more flexibility in terms of configuration management

Architecture



Shift in Operations

One key difference between the load balancers is the shift in how they are operated

Comparison	Marathon-LB	Edge-LB
Where?	Generally deployed on DC/OS public agents providing cluster-wide ingress load-balancing	Deployed on any node (public for ingress)
Who?	Application users specify labels within their app definition which are picked up by MLB from the Marathon Event Bus	Administrators deploy individual pool servers and load balancers (specific to tenants, groups of applications that share a specific configuration, or other business rules)
What?	MLB is a single container service (generally provides cluster-wide config)	3-tier service; designed to scale and provide granular control and differentiated services
How?	The service automatically generates a load balancer configuration based on the labels and reloads the HAProxy service	Applications are explicitly added to the load balancer by administrative users The load balancer pool is configured for the specific backend service by deploying a new pool configuration

Benefits of Edge-LB

- Load balances any service on DC/OS – not limited to Marathon
  - Will become critical for load balancing services deploying using the DC/OS SDK (ie. data services)
  - Will become critical for load balancing services deployed on K8s within DC/OS
- Provides much better control for cluster administrators as to what services can or should be exposed outside the cluster (ie. Internet)
- Provides more granular control over load balancer configurations
- No one size fits all configuration
- More robust solution
  - MLB could be broken by bad application specific configs

Understanding the Pool Configuration

- One of the key differences with Edge-LB from a configuration perspective is that it uses REGEX to identify front/backend services
- These are modeled in the Pool configuration file
- The following example covers a basic config
- This config can become more complex based on use case, deployment topology and the other considerations already described
- The config generates an HAProxy configuration

Sample 1: Minimal

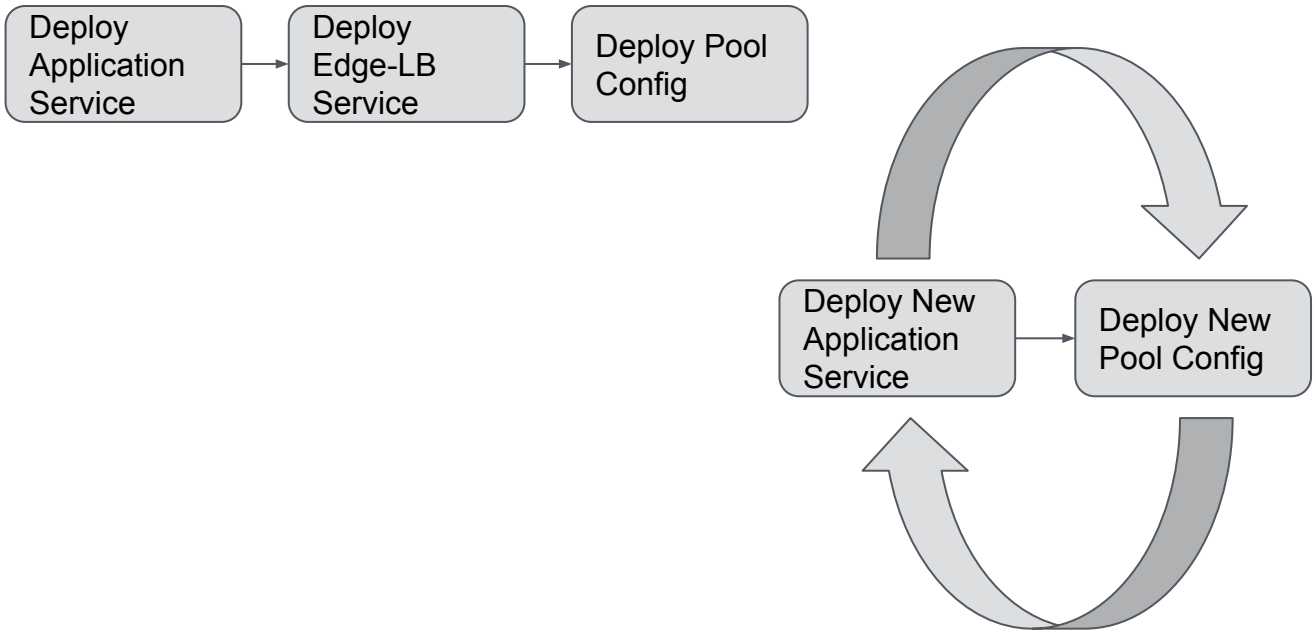
```
---
pools: - name: sample-minimal
count: 1
haproxy:
        frontends:
                - bindPort: 80
                protocol: HTTP
                linkBackend:
                        defaultBackend: svc
        backends:
                - name: svc
                protocol: HTTP
                servers:
                        - framework:
                                value: marathon
                        task:
                                value: svc-blue
                        port:
                                name: web
```

Frontend listens on Port 80

Proxy to:

Backend service with name: svc
Hosted on Marathon
Task with name: svc-blue
Port with name: web

Deployment Process



Deployment Process Command Reference

Deploy Edge-LB service

$ dcos package install --options=edge-lb-options.json Edge-LB

**Note: You will need to add the Edge-LB package repository and create the options file referenced which includes the service account details needed for the service to running securely on DC/OS

https://docs.mesosphere.com/1.10/networking/edge-lb/installing/

Deploy your Application services (these are Marathon or other apps that you want load balanced by Edge-LB)

Create your pool config file (sample1 in the previous slides as an example)

Deploy your pool config

$ dcos Edge-LB config sample-minimal.yaml

https://docs.mesosphere.com/1.10/networking/edge-lb/quickstart/

Updating the Pool Config

There a few things to consider when updating the pool config:

1. The config is idempotent - meaning you can publish a new config over the last one
2. You may use the running config as your authoritative copy
    a. In that case, you pull down the running config, update it, and post it back to the service
3. You may store the config in a configuration management system
    a. In that case you pull the config from your CMS, make the updates, check the updates in, and then post them to the service

The methods/process used to update the pool config will be based on the size of the organization, roles and responsibilities, permissions, and other factors.

Sample workflow for Updating the Pool Config

Example: Adding a new service to the pool config

1. Export the current running config to a json file

```
$ dcos edgelb config > backup.json
```

2. Add the new configuration items as needed for your new service – add corresponding `"backends"` and `"frontends"` sections.

3. Publish the new config

```
$ dcos edgelb config new_ver.json
```

4. Validate using `dcos edgelb pool config <pool name>` or just by checking service endpoints after pool is deployed.

Normal Reload Scenario

A change to a service (such as scaling up) that is load balanced by a pool will trigger a reload of its load balancers. This reload has the following properies:

No traffic is dropped (unless the service instance that was serving the request was killed).

The load balancer will wait until existing connections terminate, so a long-running connection will prevent the reload from completing.

A reload will occur at most once every 10 seconds.

The properties of this reload enable strategies like Blue/Green Deployment.

Load Balancer Relaunch Scenario

A change to the load balancer pool (such as adding a secret) will trigger a relaunch of all load balancers in the pool. This relaunch has the following properies:

Traffic is dropped
To minimize the impact, we suggest running more than one load balancer within the pool.
The load balancer will be relaunched on the same node (unless the node itself has failed).
Note: The number of instances of load balancers cannot be scaled down. This limitation will be addressed in a future Edge-LB release.

Summary

Three focus points in deciding between Marathon-LB and Edge-LB

- **Separation of Duties - How does the process fit your organization's operational model?**
  - Customer has the choice of operational model whether Ops team, dedicated Networking team, or Developers to expose applications (Controlled vs. Self-Service)
  - Edge-LB supports finer grained access control with RBAC, Secrets, and Security Accounts

- **Extensibility - What type of workloads are we focusing on?**
  - Edge-LB supports both Marathon and non-Marathon services (i.e. SDK services and frameworks)

- **Modular Nature - Fault Tolerance**
  - Edge-LB is set up to be a distributed service and is more modular in nature, more complexity but more fault tolerance
  - Marathon-LB is more monolithic in nature, less complexity but less fault tolerant
  - This allows for a load-balancing service to be "added" to a running service, rather than directly at runtime

Appendix

Advanced Load Balancing Examples

- Edge-LB: Hostname / SNI Routing with VHOSTS
- Edge-LB: Internal East/West Loadbalancing
- Edge-LB: Load Balancing Mesos Frameworks

Installing Edge-LB