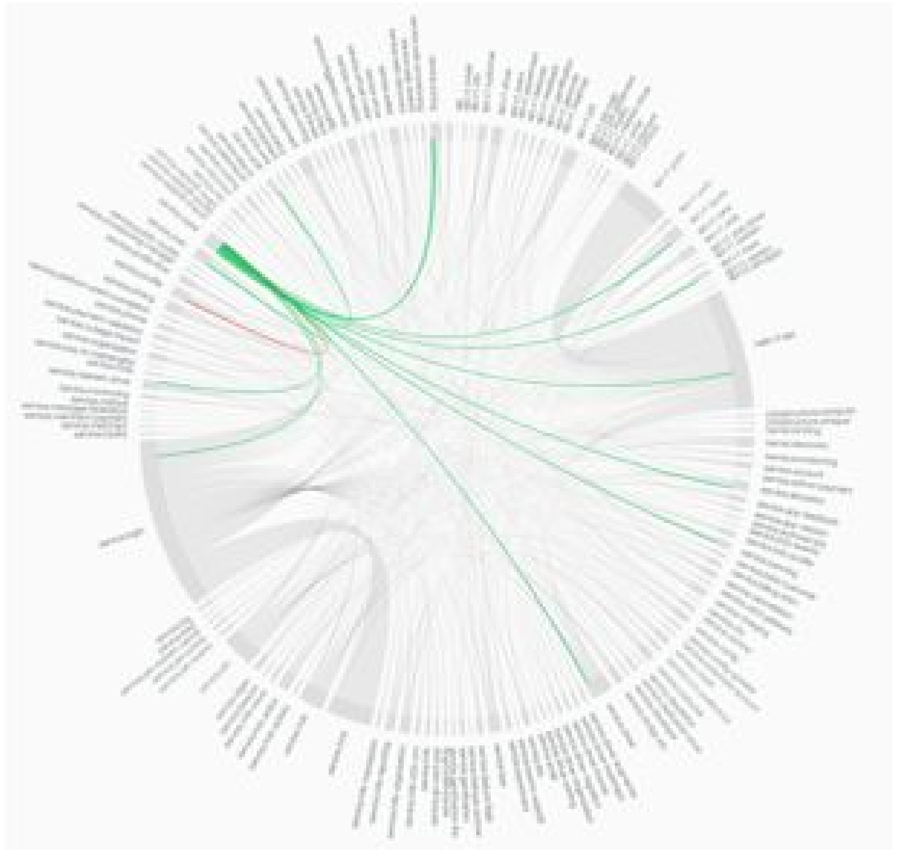
DC/OS Service Discovery

MESOSPHERE

Service Discovery

Service discovery is how your applications and services find each other.



Service Discovery in DC/OS

Mesos-DNS

Virtual IPs (VIPs)

Marathon-LB

IP-per-task

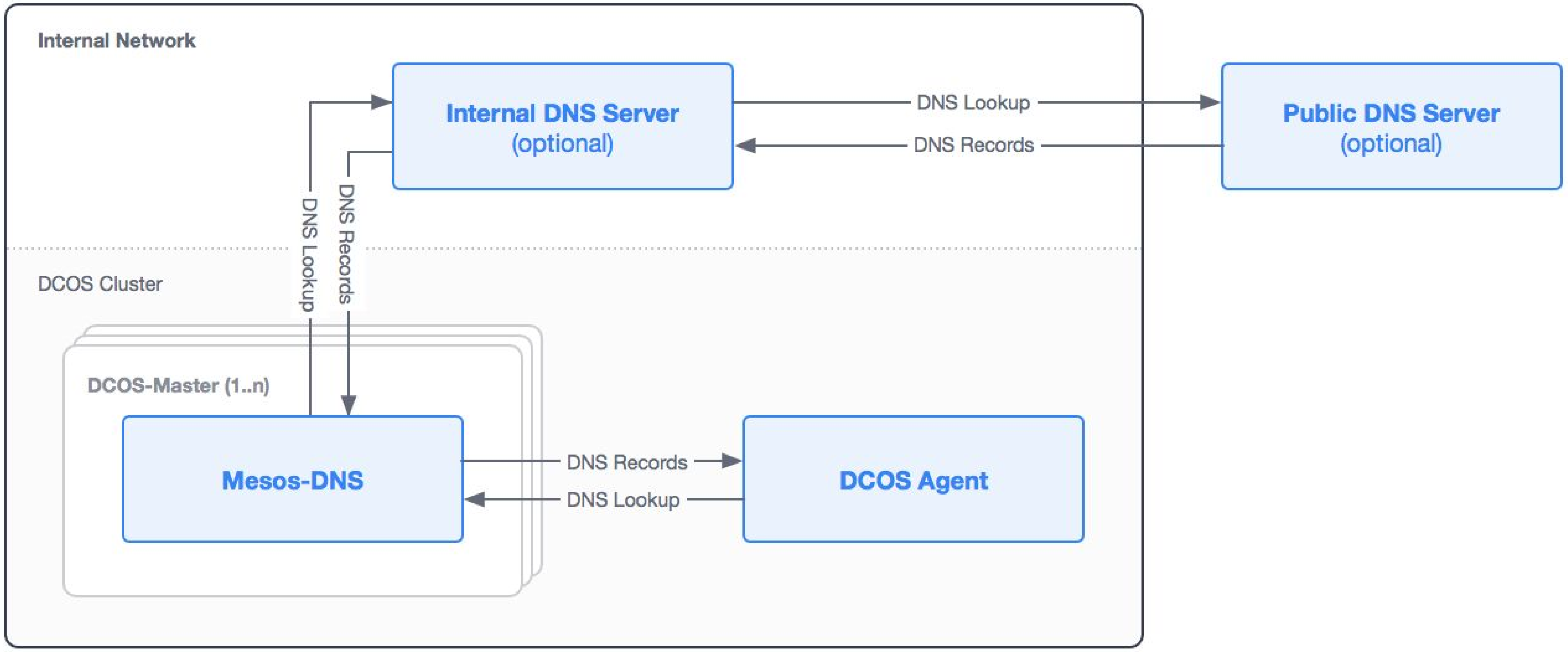
3rd Party Services

Mesos-DNS

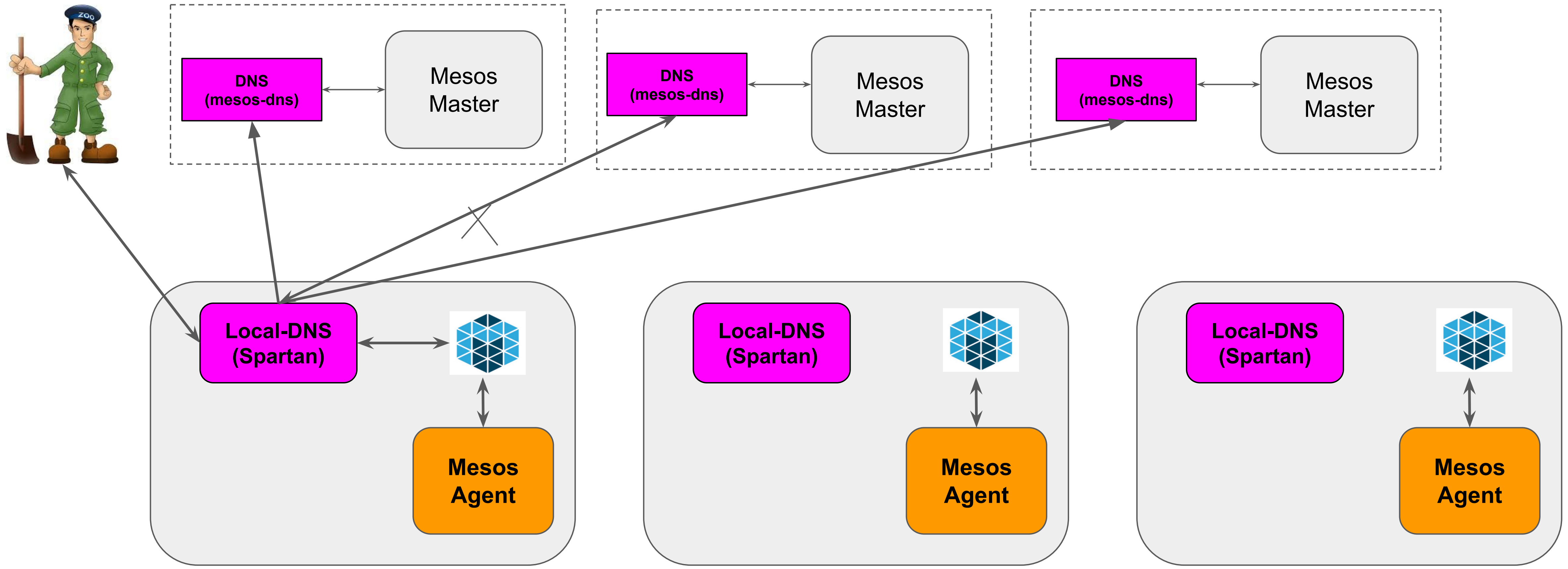
Mesos-DNS is a basic DNS-based service discovery tool that works with any Mesos task.

https://dcos.io/docs/1.9/networking/mesos-dns/

Mesos-DNS integration



Service Discovery (Mesos-DNS + Spartan)



*Spartan listens to **three** non-routable local addresses, 198.51.100.1, 198.51.100.2, 198.51.100.3*

VIPs

A layer 4 load balancer, which can be used for most TCP traffic for any Mesos task within a DC/OS cluster.

A named VIP contains 3 components:

- Private virtual IP address
- Port (a port which the service is available on)
- Service name

https://dcos.io/docs/1.9/networking/load-balancing-vips/virtual-ip-addresses/

Mesos-DNS and VIPs in Action with Kafka

```
$ dcos kafka connection
{
    "address": [
        "10.0.0.211:9843",
        "10.0.0.217:10056",
        "10.0.0.214:9689"
    ],
    "dns": [
        "broker-0.kafka.mesos:9843",
        "broker-1.kafka.mesos:10056",
        "broker-2.kafka.mesos:9689"
    ],
    "vip": "broker.kafka.l4lb.thisdcos.directory:9092",
    "zookeeper": "master.mesos:2181/dcos-service-kafka"
}
```

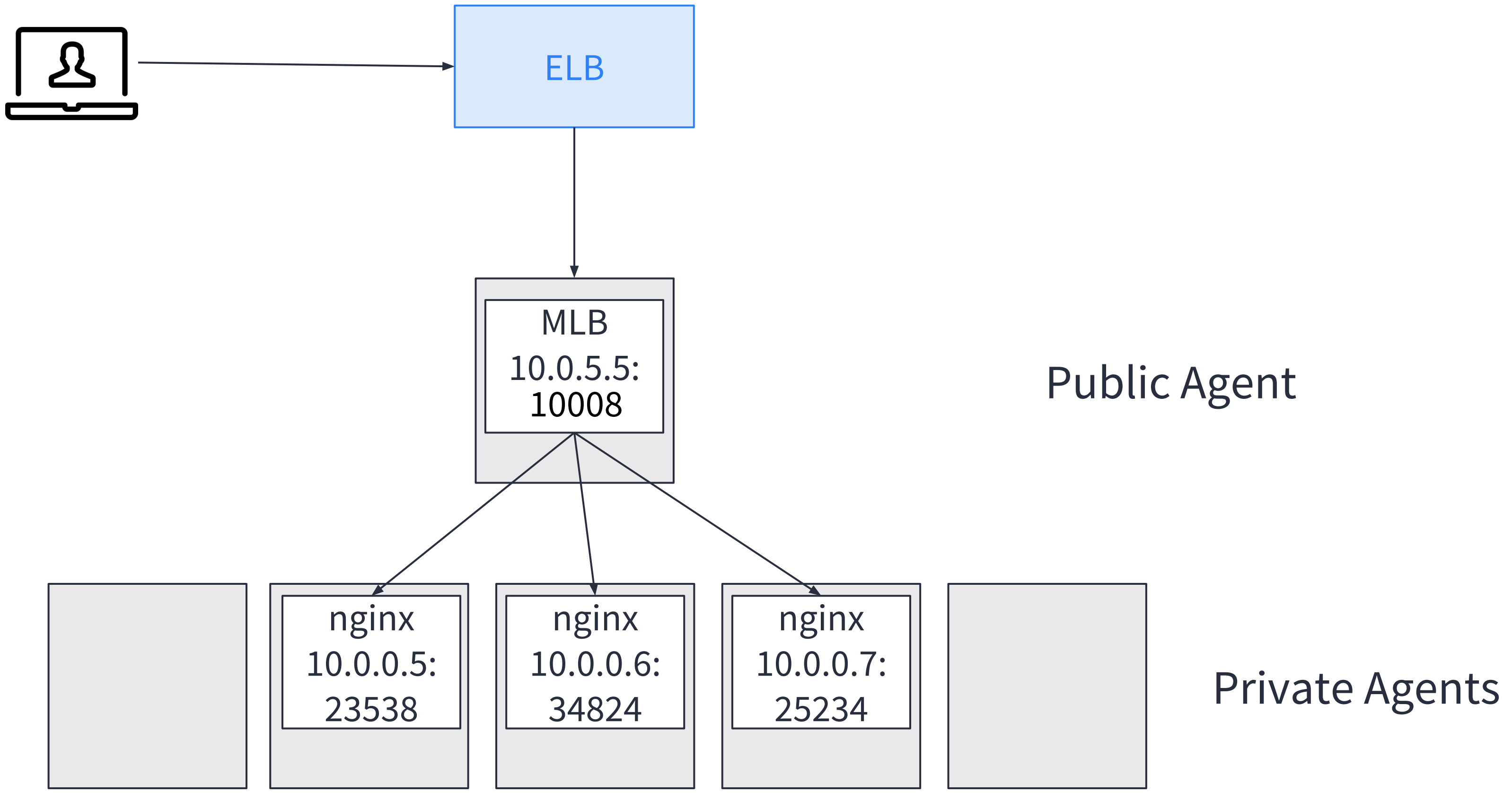
Marathon-LB

Marathon-LB is an HAProxy-based load balancer for Marathon only.

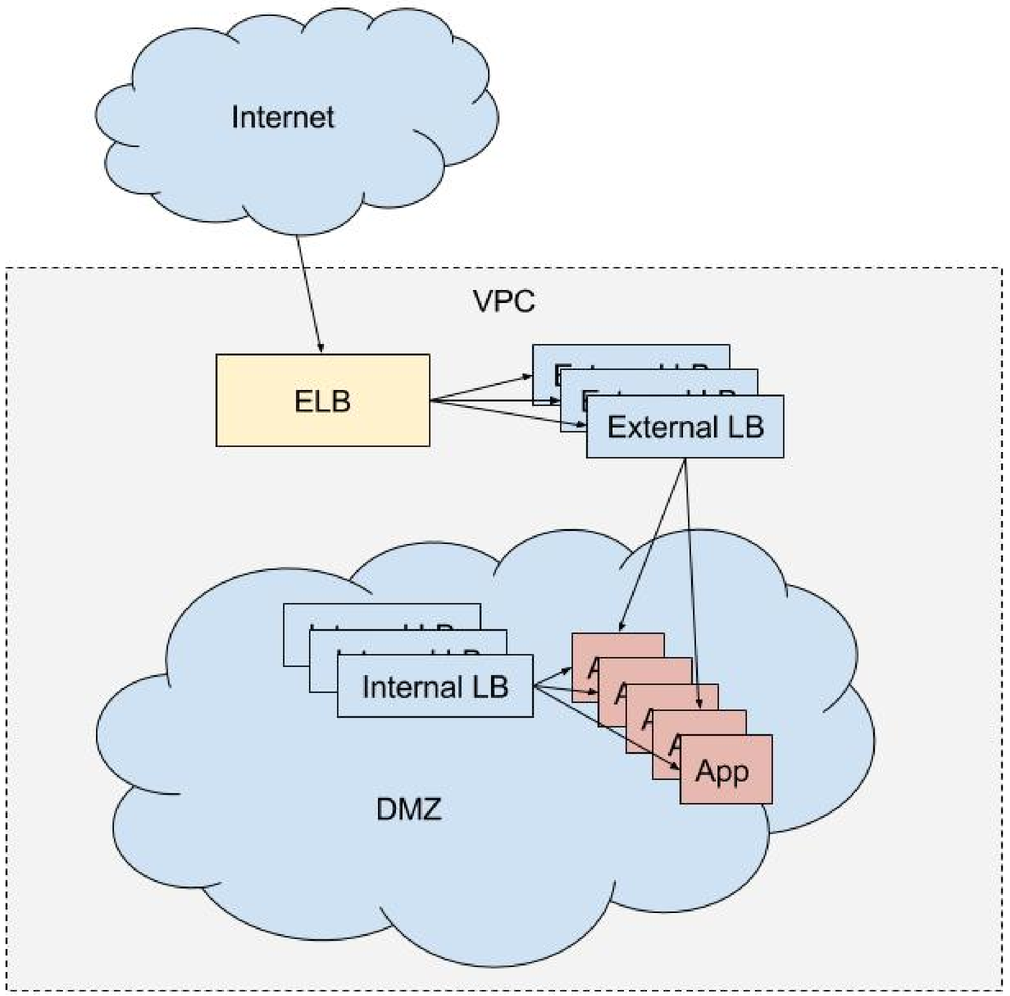
https://dcos.io/docs/1.9/networking/marathon-lb/

MARATHON-LB LAB

```
"id": "nginx",
“instances”:3,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image":
"nginx:1.7.7",
      "network": "BRIDGE",
      "portMappings": [
        { "hostPort": 0,
"containerPort": 80,
"servicePort": 10008 }
      ],
```



Marathon-LB as an internal and external load balancer



3rd Party Service Discovery: linkerd

linkerd is a service mesh for cloud-native applications: https://linkerd.io/

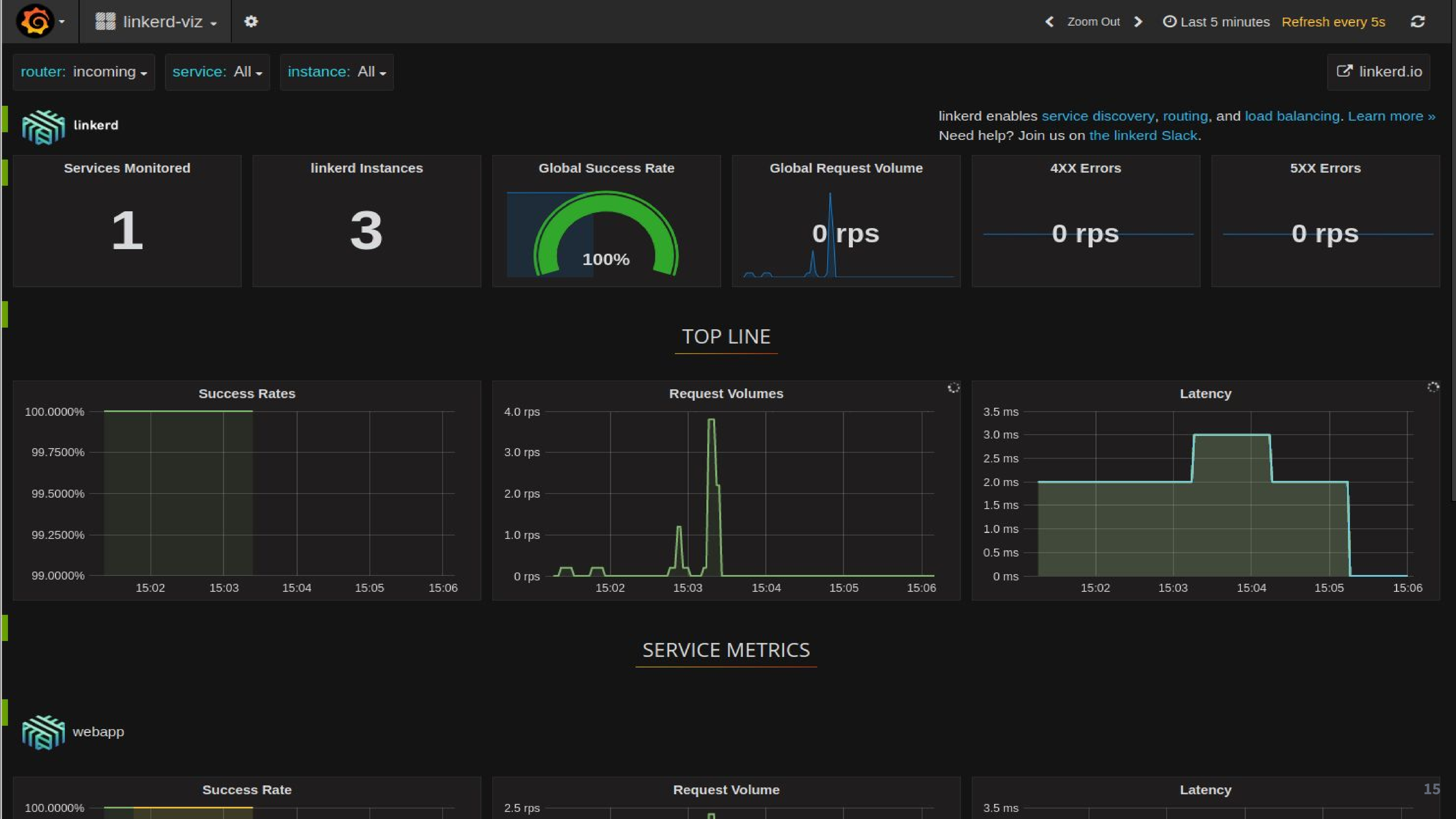
“takes the name of a service and of a call to make on that service (HTTP, gRPC, etc.), and does the work required to make the call successful—including routing, load-balancing, and retrying.”

Linkerd on DC/OS

DC/OS has linkerd (installed per node) and linkerd-viz (a single service installed for metrics) packages in the Universe catalog.

Every agent gets linkerd installed, a single instance of linkerd-viz may also be installed for metrics.

Applications can use their node-local linkerd instance to send traffic through the service mesh and take advantage service discovery, resilient communication, and top-line service metrics.



router: incoming service: All instance: All

linkerd enables service discovery, routing, and load balancing. Learn more »
Need help? Join us on the linkerd Slack.

3rd Party Service Discovery: Linkerd Resources

Resources

- https://linkerd.io/getting-started/dcos/
- https://github.com/dcos/examples/tree/master/linkerd/
- https://dcos.io/blog/2016/service-discovery-and-visibility-with-ease-on-dc-os/index.html

Usage

When to use what?

VIPs	Marathon-LB	MesosDNS
Distributed, L4, Scalable, TCP Only	External, internal L7 traffic (TLS termination, zero-downtime deployments, HTTP sticky sessions),	marathon-lb.marathon.mesos UDP services, SRV vs A records

How Redis benefits from DC/OS Service Discovery

Mesos-DNS A record automatically assigned:

```
    redis.marathon.mesos
```

Includes an SRV record which includes the port (:

```
    $ dig srv _redis._tcp.marathon.mesos
    ;; ANSWER SECTION:
    _redis._tcp.marathon.mesos. 60  IN  SRV 0 0 30585 redis-1y1hj-s1.marathon.mesos.

    ;; ADDITIONAL SECTION:
    redis-1y1hj-s1.marathon.mesos. 60 IN  A 10.0.0.43
```

VIP for Redis may look like: `redis.marathon.l4lb.thisdcos.directory:6379`

101 Tutorial: Connecting Apps/Service Discovery, with Redis example: *https://dcos.io/docs/1.9/tutorials/dcos-101/service-discovery/*

Redis example deployment documentation: *https://github.com/dcos/examples/tree/master/redis*