Infoblox
NEXT LEVEL NETWORKING

White Paper

CoreDNS as DNS Resolver for Microservices deployed in Hybrid Clouds

May 14th 2019

By:
Shankar Ganesh PJ
Staff Engineer, Cloud & Nex Gen Team.

TABLE OF CONTENTS

Summary

CoreDNS has become a popular DNS server for cloud native deployments. Kubernetes version 1.13 officially supports CoreDNS as the default DNS server. As cloud native deployment is gaining prominence, customers are expected to face minor issues while deploying their applications as services over containers.

Services are primarily identified by a host name associated with an IP address, and the resolution is brought by DNS services running with kubernetes. However, the scope of such DNS services are within individual control plane. Management of customer workloads deployed across a hybrid or multi-cloud environment using microservice architecture is a challenge to customers, as there is no single pane of glass for complete DNS data.

These challenges must be addressed effectively. If the customers have clusters spanning multiple clouds, then they should manage their environment using external DNS solution. This opens up new opportunities for Infoblox, such as managing CoreDNS through Infoblox's core network services. This white paper explains how this works

CoreDNS Overview

CoreDNS is an open-source DNS server that supports DNS-based service discovery in an environment that uses containers, including Docker containers written in Go that chains plugins. It can be used in a multitude of environments because of its flexibility. Each plugin performs a DNS function, such as Kubernetes service discovery, Prometheus metrics or rewriting queries and many more.
The following list highlights what CoreDNS is:

- CoreDNS is a Cloud Native Computing Foundation graduated project.
- CoreDNS is a fast and flexible DNS server. The key word here is flexible as CoreDNS allows you to do what you want with your DNS data by utilizing plugins.
- CoreDNS can listen for DNS requests coming in over UDP/TCP, TLS (RFC 7858), also known as DoT, DNS over HTTP/2 - DoH - (RFC 8484) and gRPC (not a standard).
- CoreDNS acts as the default DNS service in k8s.
- In Kubernetes 1.13, CoreDNS is the default cluster DNS server.

Challenges

The following are some common issues faced by customers after deploying services using Kubernetes:

- No unified visibility of the microservices running on the cluster.
- Containers/services/pods are currently ran in a cluster. This makes it difficult to see all the details in a single pane with native Kubernetes commands, as the results are cumbersome.
- It is easy to handle and see what is running in the cluster on a small scale. However, as the scale increases, a tool/service is required to give us better visibility.
- CoreDNS running as a default pod cannot be exposed to the outside world using a public IP.

Solution

Infoblox is solving the common issues faced by customers by integrating CoreDNS with CoreDNS with NIOS (Network Identity Operating System). NIOS is the operating system that powers Infoblox's core network services, ensuring non-stop operation of network infrastructure. As the base for next level networking, NIOS automates the error-prone and time-consuming manual tasks associated with deploying and managing DNS, DHCP, and IP address management (IPAM) required for continuous network availability and business uptime[2].

Note: NIOS can be either on-prem or in any public clouds.

The following are the configurations needed for a solution to work in various cloud environments:

- A NIOS Grid with DNS servers.(one for external and one for internal)
- A Kubernetes cluster on any platform (up and running).
- A CoreDNS pod deployed in the cluster with CoreFile including both plugin "kubernetes" and "k8s_external" (must be a CoreDNS image >= 1.3.0) - this "CoreDNS service" need to be visible from NIOS DNS Servers. That means opening up the service on a Load Balancer.
- Gathering the IP of the service CoreDNS running on the cluster.
- Using CoreDNS as a default DNS resolver, (currently even K8s has made CoreDNS as the default).
- Making the CoreFile to create zones such as "Internal" and "External" zones.

- The "**internal-view**" (the Service IP is cluster IP, and the FQDN is part of the cluster.local zone of the cluster)
- The "**external-view**" (the Service IP it its LoadBalancer IP, and fqdn is part of an external zone to build up)

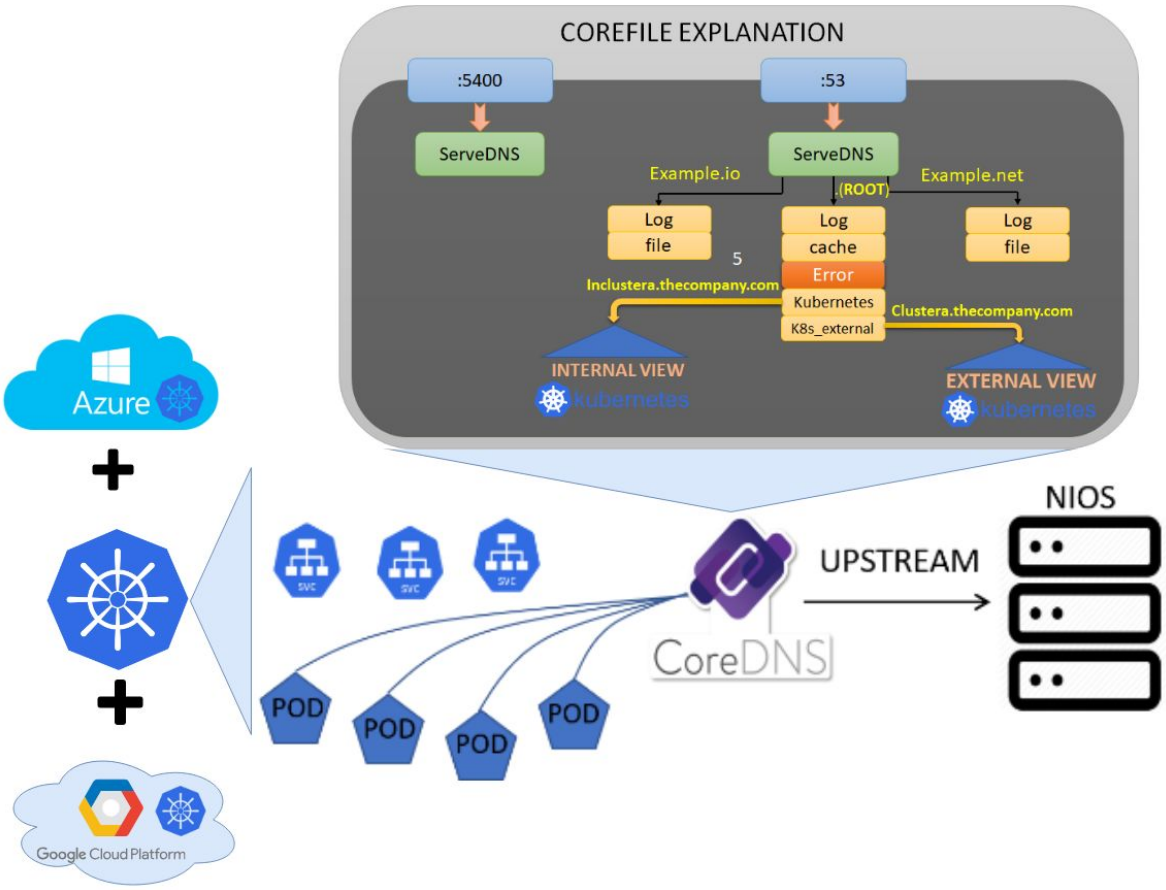

Fig: An Overview of how CoreDNS works with Infoblox's Configured CoreFile

```
Corefile: |
   .:53 {
      kubernetes internalcluster.thecompany.com in-addr.arpa
ip6.arpa {
         transfer to *
         upstream
         fallthrough in-addr.arpa ip6.arpa
      }
      K8s_external externalcluster.thecompany.com {
```

```
        transfer to *
    }
```

Conclusion

Using CoreDNS along with the Infoblox services provides a complete end-to-end solution with the following advantages:

- Visibility of the Kubernetes services in NIOS
- Quick integration with external DNS solutions
- Complete visibility across hybrid clouds
- Dedicated support

References

CoreDNS https://coredns.io/
Infoblox Solutions for Cloud[1] - https://www.infoblox.com/solutions/cloud-computing/
https://community.infoblox.com/t5/Company-Blog/SaaS-What-s-In-It-for-You/ba-p/11252

NIOS reference link[2]
https://www.infoblox.com/glossary/network-identity-operating-system-nios/