BUILDING A KUBERNETES OPERATOR TO INTEGRATE WITH CERN'S DNS API

September 2023

AUTHOR(S):
Karan Mishra

Thapar University, Patiala, India

SUPERVISOR(S):
Jack Henschel

CERN openlab

ABSTRACT

In the rapidly evolving digital landscape of today, effective management of DNS (Domain Name System) configurations has become a critical task for organizations of all sizes. One of the key challenges faced by these organizations is efficiently handling delegated domains, a task that requires automation and streamlining. To address this challenge, the landb-operator project was conceived. This project not only introduces a Kubernetes Operator but also offers a versatile command-line interface (CLI) tool, providing a comprehensive solution to the complexities associated with managing delegated domains.

DNS delegation stands as a pivotal component of modern network infrastructure, allowing organizations to assign responsibility for specific subdomains to different DNS servers. This process is indispensable for maintaining a coherent and organized web presence. In practice, organizations frequently find themselves tasked with overseeing a multitude of delegated domains, each with its unique set of records and configurations. These domains are distributed across various services and applications, and managing them can quickly become a formidable challenge.

The landb-operator project steps in to simplify and streamline this intricate domain management process within Kubernetes environments. Kubernetes, known for its robust container orchestration capabilities, serves as the ideal platform to house this innovative solution. The primary objective of this project is to provide comprehensive toolset for managing delegated domains at CERN seamlessly. At the heart of this project is the concept of synchronization, ensuring that changes made within the Kubernetes cluster are accurately reflected in external DNS services. CERN's DNS SOAP (Simple Object Access Protocol) API is a notable example of an external DNS service that can be seamlessly integrated with the landb-operator. This integration is crucial for ensuring that DNS records and configurations remain consistent, regardless of where the changes originate.

PROJECT SPECIFICATION

The primary goal of the landb-operator project is to offer a robust solution for managing delegated domains as custom resources within a Kubernetes cluster. This approach enables organizations to leverage Kubernetes' powerful orchestration capabilities for DNS management. The project's core objectives include:

• Custom Resource Management: The landb-operator introduces a custom resource definition (CRD) called "delegated domain" to Kubernetes. This CRD serves as a representation of delegated domains and allows users to create, update, delete, and query delegated domain configurations as if they were native Kubernetes objects.

• Synchronization: Delegated domains often require synchronization between the Kubernetes cluster and external DNS providers. The landb-operator ensures that changes made to delegated domain resources within Kubernetes are reflected in the external DNS system, such as CERN's DNS SOAP API. This synchronization guarantees the accuracy and consistency of DNS configurations.

• Automation: By leveraging Kubernetes Operators, the landb-operator automates routine DNS management tasks, reducing the operational overhead associated with manual configuration changes. This automation enhances the efficiency of managing delegated domains.

Components: The landb-operator project comprises three key components:

• landb-operator: This Kubernetes Operator manages the lifecycle of delegated domain resources within the cluster. It listens for changes to delegated domain CRDs, executes reconciliation logic, and ensures that the desired state of delegated domains is maintained. The operator interacts with external DNS services to propagate changes when necessary.

• landb-cli: The command-line interface tool complements the landb-operator by providing administrators with a convenient way to interact with delegated domains. Users can create, update, delete, and query delegated domain configurations using intuitive CLI commands. This tool is particularly useful for manual operations and scripting.

• landb-helm: To simplify the deployment of the landb-operator within Kubernetes clusters, Helm charts have been created. These charts encapsulate best practices for configuring and scaling the operator, making it easier for administrators to get started with the landb-operator.

PLATFORM AS A SERVICE (PAAS)

Platform as a Service (PaaS) is a cloud computing service model that provides a platform and environment for developers to build, deploy, and manage applications without having to worry about the underlying infrastructure and hardware. In essence, PaaS offers a set of tools, services, and resources that facilitate the development, testing, and deployment of software applications. Key characteristics of Platform as a Service include:

• Abstraction of Infrastructure: With PaaS, developers can focus primarily on writing code and building applications rather than dealing with the complexities of setting up and managing servers, storage, networking, and other infrastructure components. The underlying hardware and software infrastructure are abstracted and managed by the PaaS provider.

• Development Tools: PaaS offerings typically come with a range of development tools, libraries, and frameworks that make it easier for developers to create applications. These tools might include programming languages, runtime environments, databases, and middleware.

• Scalability: PaaS platforms often include built-in scalability features that allow applications to automatically scale up or down based on demand. This means that as the number of users or transactions increases, the platform can allocate additional resources to ensure optimal performance.

• Deployment and Management: PaaS providers offer tools for deploying applications to the cloud, managing the application's lifecycle, and handling tasks like version control, testing, monitoring, and updates.

• Multi-Tenancy: PaaS platforms are typically multi-tenant environments, meaning that multiple users or organizations can use the same underlying infrastructure while maintaining isolation and security for their individual applications and data.

PaaS AT CERN

OKD4 (OpenShift 4) provides a platform to host user applications in a self-service fashion, following the Platform-as-a-Service approach. This service is intended to facilitate application deployment by removing the need for application owners to manage the computing infrastructure hosting the application while providing a high level of automation and integration with the CERN computing environment.

OKD4 is a distribution of the popular Kubernetes container orchestration platform, with a focus on multi-tenancy: the ability to host many applications isolated from each other in a common environment.

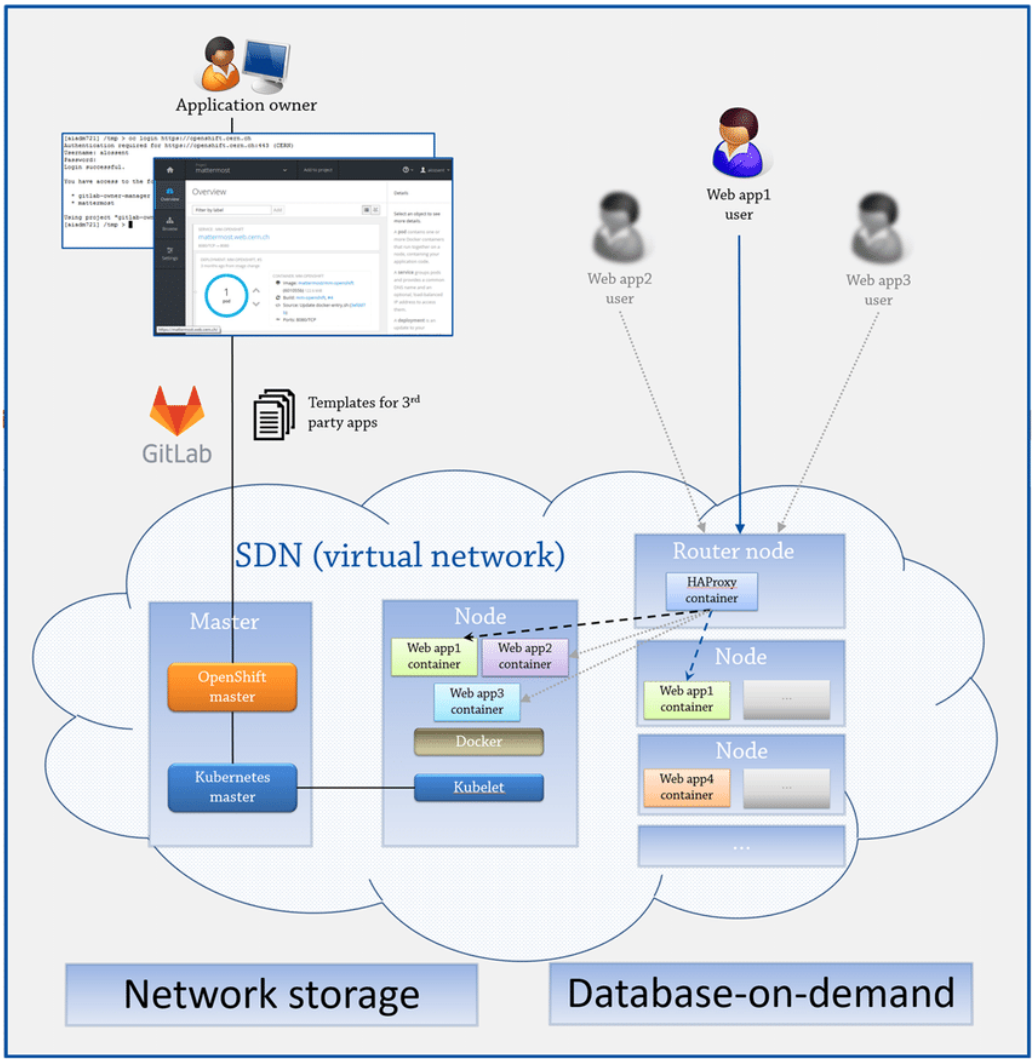

Figure 1: OpenShift PaaS for web applications service architecture

LANDB OPERATOR

1 DELEGATED DOMAIN

Delegated domains refer to a situation in the Domain Name System (DNS) hierarchy where the responsibility for managing and maintaining a specific subdomain is delegated to a different organization or DNS server. This delegation allows for the distribution of DNS management across multiple entities, making it easier to manage large and complex domain name systems.

Here's how delegated domains work:

• Top-Level Domain (TLD): The DNS hierarchy begins with top-level domains (TLDs) like .com, .org, .net, .gov, and country-code TLDs like .ch, .in, etc. TLDs are managed by domain registrars and are the highest level of the DNS hierarchy.

• Second-Level Domain (SLD): Below the TLD, you have second-level domains (SLDs). These are the domain names that individuals and organizations register and use. For example, in the domain cern.ch, "cern" is the second-level domain.

• Subdomains: Under second-level domains, you can create subdomains. A subdomain is a domain that is part of a larger domain but is managed separately. For instance, if you have cern.ch, you can create subdomains like docs.cern.ch or cds.cern.ch. Subdomains can be used to organize websites and services.

Delegation: When you create a subdomain and want to manage its DNS records separately from the parent domain, you can delegate the subdomain to a different DNS server or provider. This is done by specifying the DNS servers responsible for resolving the subdomain's DNS records in the parent domain's DNS settings.
For example, if you want to delegate docs.cern.ch to a different DNS provider, you would configure the DNS records for docs.cern.ch on that provider's DNS servers and update the DNS records for cern.ch to point to the DNS servers of the delegated provider for the subdomain docs.cern.ch.
Delegated domains are commonly used in scenarios where different departments or organizations need control over their subdomains while maintaining a cohesive top-level domain. It allows for distributed management and customization of DNS settings for various services or websites under the same parent domain.

2 DNS API AT CERN

CERN's DNS API is a powerful tool for managing domain name services within the CERN network infrastructure. It provides a programmatic interface to interact with DNS services, allowing users to perform operations like domain delegation, record management, and more. With this API, users can automate DNS-related tasks, ensuring efficient and accurate management of domain resources across the CERN ecosystem.

• DNS Delegated Search: Method used to look for a delegated domain already registered in the system.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:NetworkService">
```

```
   <soapenv:Header>
      <!-- Authentication Token in Header -->
      <urn:AuthToken>YOUR_AUTH_TOKEN</urn:AuthToken>
   </soapenv:Header>
   <soapenv:Body>
      <!-- Body of the Request -->
      <urn:dnsDelegatedSearch>
         <!-- Search Text String -->
         <urn:Search>example-search-text</urn:Search>
      </urn:dnsDelegatedSearch>
   </soapenv:Body>
</soapenv:Envelope>
```

• DNS Delegated Add: Method used to add a new delegated domain to the infrastructure. It takes parameters such as domain name, view, keyname, description, and user description.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:NetworkService">
   <soapenv:Header>
      <!-- Authentication Token in Header -->
      <urn:AuthToken>YOUR_AUTH_TOKEN</urn:AuthToken>
   </soapenv:Header>
   <soapenv:Body>
      <!-- Body of the Request -->
      <urn:dnsDelegatedAdd>
         <!-- Domain Information -->
         <urn:domain>example.com</urn:domain>
         <!-- View: Internal or External -->
         <urn:view>Internal</urn:view>
         <!-- Keyname for Delegation -->
         <urn:keyname>example-key</urn:keyname>
         <!-- Description -->
         <urn:description>This is a delegated domain.</urn:description>
         <!-- User Description -->
         <urn:userDescription>Additional information from the user.</urn:userDescription>
      </urn:dnsDelegatedAdd>
   </soapenv:Body>
</soapenv:Envelope>
```

• DNS Delegated Remove: Method used to remove a registered delegated domain from the infrastructure. It takes parameters domain name and view.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:Netwo
   <soapenv:Header>
      <!-- Authentication Token in Header -->
      <urn:AuthToken>YOUR_AUTH_TOKEN</urn:AuthToken>
   </soapenv:Header>
   <soapenv:Body>
      <!-- Body of the Request -->
      <urn:dnsDelegatedRemove>
```

```
        <!-- Domain to Remove -->
        <urn:Domain>example-domain.com</urn:Domain>
        <!-- View (Internal/External) -->
        <urn:View>Internal</urn:View>
      </urn:dnsDelegatedRemove>
   </soapenv:Body>
</soapenv:Envelope>
```

3 KUBERNETES OPERATORS

Kubernetes, as a container orchestration tool, offers limited initial functionality to ensure automation and scalability. Kubernetes Operators are software extensions that make use of Kubernetes APIs to extend its behavior. Operators use Custom Resource Definitions (CRDs) to extend Kubernetes with custom resources that represent your application so that the operator knows how to manage it according to your application's requirements. One major benefit of operators is providing automation in stateful applications. Kubernetes Operators automate routine tasks involved in running and managing applications, such as configuration updates, backups, scaling, and failover. This reduces manual intervention and the potential for human error.

4 LANDB OPERATOR

A. CRDS AND CONTROLLER

The landb-operator is a critical component designed for the efficient management of a custom resource known as delegated domain within a Kubernetes environment. This custom resource, delegated domain, is instrumental in keeping a record of delegated domain instances. The primary purpose of this operator is to enable the monitoring and synchronization of these domain delegations across the Kubernetes cluster where it is deployed. Additionally, it orchestrates the propagation of corresponding CRUD (Create, Read, Update, Delete) operations to the CERN's DNS SOAP API, ensuring coordination between the Kubernetes cluster and the external DNS infrastructure.
Key Features and Functionalities:

• Custom Resource Management: The landb-operator is responsible for the creation, modification, retrieval, and deletion (CRUD) operations related to the delegated domain custom resource. This allows Kubernetes users to effortlessly manage delegated domains within their cluster.

• Real-time Synchronization: Whenever a CRUD operation is executed on a delegated domain, the landb-operator ensures that the changes are instantly reflected within the Kubernetes cluster where it is deployed. This real-time synchronization guarantees that the cluster's view of the domain delegation remains up-to-date.

• Integration with CERN's DNS SOAP API: To maintain consistency between the Kubernetes cluster and CERN's DNS infrastructure, the operator communicates with the CERN DNS SOAP API. This integration enables the operator to send corresponding requests to the API whenever a CRUD operation occurs on a delegated domain.

• Error Handling and Resilience: The landb-operator is designed to handle errors gracefully. In cases of failed communication with the DNS API or any other errors, it employs robust error handling mechanisms to ensure the stability and reliability of the domain delegation management.

• Logging and Monitoring: To facilitate debugging and monitoring, the operator maintains comprehensive logs of all operations. This assists administrators in tracking changes, diagnosing issues, and ensuring the smooth operation of the domain delegation management system.

• Scalability: The landb-operator is engineered for scalability, making it suitable for both small and large-scale Kubernetes clusters. It efficiently manages a growing number of delegated domains as the cluster expands.

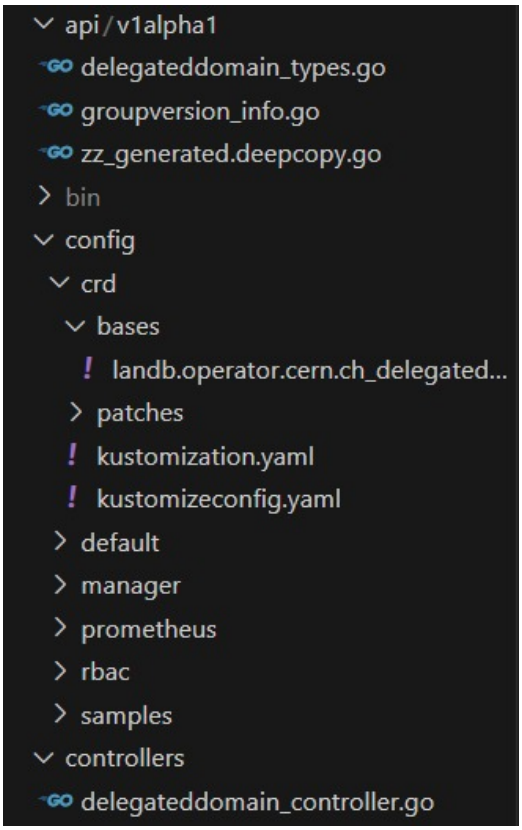

Figure 2: landb-operator directory structure

The Landb Operator consists of a structured directory layout, including a bin folder containing binaries, an api/v1alpha1 directory with manifest files for generating Custom Resource Definitions (CRDs), and a config/crds directory for storing CRDs in YAML format. The operator also includes a controller directory, which houses the logic for handling operations.

The description of a delegated domain looks as follows and includes all fields required to register a domain in the infrastructure:

```
type DelegatedDomainSpec struct {
        // Important: Run "make" to regenerate code after modifying this file

        Description     string   `json:"description,omitempty"`
        Domain          string   `json:"domain,omitempty"`
        Keyname         string   `json:"keyname,omitempty"`
        View            ViewType `json:"view,omitempty"`
        UserDescription string   `json:"userDescription,omitempty"`
}
```

The controller file consists of a reconcile function with the following outline:

```
func (r *DelegatedDomainReconciler) Reconcile(ctx context.Context, req ctrl.Request)
(ctrl.Result, error){}
```

This function contains the entire logic of the operator and sends requests to DNS API.

B. COMMAND LINE INTERFACE

In addition to the landb-operator, we have developed a command-line tool using the Cobra library to provide a manual interface for managing delegated domains within a Kubernetes environment. This command-line tool serves as a versatile and user-friendly utility that enables administrators to perform domain-related operations swiftly and efficiently.

Key Features and Functionalities:

• Command-Line Interface (CLI): The tool offers a command-line interface that is intuitive and easy to use. Users can interact with the tool by executing simple commands in their terminal or shell.

• CRUD Operations: Just like the landb-operator, the CLI tool allows users to perform CRUD (Create, Read, Update, Delete) operations on delegated domains. This includes creating new delegated domains, modifying existing ones, retrieving domain information, and removing domains.

• Real-time Updates: All operations performed through the CLI tool are executed in real-time. This means that changes made to delegated domains take effect immediately, ensuring that the Kubernetes cluster's domain management is always up-to-date.

• Interactive Prompts: To streamline the user experience, the tool incorporates interactive prompts when necessary. These prompts guide users through the domain management process, seeking required input and providing feedback along the way.

• Error Handling: Robust error handling is a core component of the CLI tool. It provides informative error messages and guidance in cases of incorrect inputs (flags) or failed operations, helping users troubleshoot issues effectively.

• Documentation: Comprehensive documentation accompanies the CLI tool, providing users with clear instructions and examples for performing various domain operations. This documentation serves as a valuable resource for both new and experienced users.

More details about the SOAP API endpoint can be referred to here https://network.cern.ch/sc/soap/6/

C. HELM CHARTS

In the context of managing the landb-operator and ensuring seamless deployments to Kubernetes clusters, we've leveraged Helm charts as a powerful tool for packaging, versioning, and deploying Kubernetes applications.
Helm is a package manager for Kubernetes applications. It allows us to define, install, and upgrade even the most complex Kubernetes applications. Helm charts provide a higher level of abstraction, making it easier to manage resources, dependencies, and configurations.
The landb-operator is a crucial component for managing delegated domains within our infrastructure. To streamline its deployment, we've created Helm charts tailored specifically to the needs of the operator.

Key Components of the Helm Charts:

• Chart Structure: The Helm chart for the landb-operator is structured into directories, with Chart.yaml defining metadata, values.yaml containing default configuration values, and templates holding Kubernetes resource manifests.

• Customization: We've designed the Helm charts to be highly customizable. Users can modify values in the values.yaml file to adapt the operator's behavior to their specific requirements.

• Dependency Management: Helm allows us to define dependencies on other charts. This feature is valuable for incorporating related tools or resources into the landb-operator deployment.

• Version Control: Helm charts are versioned, enabling us to track changes and manage upgrades effectively. This ensures consistency across different deployments.

• Templating: Helm employs Go templating to generate Kubernetes manifests. This enables dynamic resource creation, making it easier to adapt to different environments or scenarios.

In summary, Helm charts are instrumental in deploying and managing the landb-operator in Kubernetes clusters. They enhance deployment reliability, maintainability, and flexibility, enabling us to efficiently manage delegated domains and ensure the smooth operation of our infrastructure.

FUTURE SCOPE

• Support for DDA: Delegated Domain Alias (DDA) typically refers to an additional domain name that is associated with or points to the primary delegated domain. This concept is often used in DNS (Domain Name System) configurations and web hosting.
For instance, if "example.com" is the primary delegated domain, a delegated domain alias like "example.net" or "example.org" could be set up to redirect users to the same website hosted at "example.com." This ensures that users who enter different domain names still end up at the same online destination.

• Incorporating other API methods: Currently the operator and the cli have support for methods like dnsDelegatedAdd, dnsDelegatedSearch, and dnsDelegatedRemove to support the basic functionality. There are other methods available, for ex. dnsDelegatedGetByNameView and dnsDelegatedListKeys that can be later introduced as the complexity of the operator increases.

CONCLUSION

The landb-operator project offers a comprehensive solution for managing delegated domains within Kubernetes environments. By introducing custom resources, synchronization mechanisms, and automation, it simplifies the often complex task of DNS management. The accompanying CLI tool provides flexibility and control to administrators, while Helm charts ease deployment. This project aims to enhance the operational efficiency and reliability of DNS management in modern, dynamic infrastructures.
As DNS continues to play a critical role in network operations, the landb-operator project stands as a valuable tool for organizations seeking to streamline delegated domain management in Kubernetes clusters.

Over the summer, the project successfully accomplished its goal of creating a functional landb-operator and a manual interface in the form of a command line tool. However, there are some parts which could be further worked upon. Working on this project taught me a lot about IaC (Infrastructure as Code), DNS service, and cloud-native architecture and helped me write better quality code in GoLang which I couldn't have done without the guidance of my supervisor Jack. I am grateful to the CERN IT-PW department for providing me with this opportunity of a lifetime and hope to keep contributing to the team in the future.