Hochschule für Berufstätige Darmstadt

– Fachbereich Informatik –

Automatische DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur

Abschlussbericht zur

Berufspraktischen Phase

vorgelegt von

Karsten Siemer

Hempberg 1
22848, Norderstedt
Matrikelnummer: 906507

Referent	:	Herr Prof. Dr. Jürgen Kühnlein
Abgabetermin	:	17.01.2024

Die berufspraktische Phase wurde in der Zeit vom 01.10.2023 bis zum 17.01.2024 bei der *SDA SE Open Industry Solutions* in Hamburg durchgeführt.

EHRENERKLÄRUNG

Ich, Karsten Siemer, versichere durch meine Unterschrift, dass ich die vorliegende Arbeit selbstständig erstellt habe. Andere als die angegebenen Hilfsmittel habe ich nicht verwendet.

Soweit ich fremde Gedankengänge oder Texte verwendet habe, sind diese von mir als solche kenntlich gemacht und dem Urheber eindeutig zuordenbar. Dazu zählen sowohl wörtliche als auch nicht wörtliche Übernahmen.

Norderstedt, 12.12.2023

Ort, Datum

Karsten Siemer

ABSTRACT

The focus of this work is on the automatic management of DNS zones and entries in a distributed multi-cloud microservices architecture. The goal is to relieve developers from manual tasks and enhance efficiency. The work includes the selection and implementation of a Kubernetes operator, the conceptualization of a DNS hierarchy, the security of public and private zones, the implementation of DNSSEC, and an infrastructure translation using Terraform. The project follows an iterative model, with weekly meetings ensuring organization. The results meet the requirements, have been successfully tested, and achieve the break-even point after four days, emphasizing the cost-effectiveness of the solution. The impact on developers is positive, as they can now deploy microservices on various cloud providers without manual DNS configuration. The conclusion demonstrates that automating the DNS infrastructure is a significant step towards efficiency and security.

ZUSAMMENFASSUNG

Diese Arbeit befasst sich mit der automatischen DNS-Zonen- und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur. Das Ziel besteht darin, die Entwickler von manuellen Tätigkeiten zu entlasten und die Effizienz zu steigern. Die Arbeit umfasst die Auswahl und Implementierung eines Kubernetes Operators, sowie die Konzeption einer DNS-Hierarchie, die Sicherheit von öffentlichen und privaten Zonen, die Implementierung von DNSSEC und eine Infrastruktur-Übersetzung mittels Terraform. Der Projektverlauf folgt einem iterativen Modell, wobei wöchentliche Meetings die Organisation sicherstellen. Die Ergebnisse entsprechen den Anforderungen, wurden erfolgreich getestet und erreichen den Break-Even-Point nach vier Tagen, was die Wirtschaftlichkeit der Lösung unterstreicht. Die Auswirkungen auf die Entwickler sind positiv, da sie nun Microservices auf verschiedenen Cloud-Providern ohne manuelle DNS-Konfiguration bereitstellen können. Die Bilanz zeigt, dass die Automatisierung der DNS-Infrastruktur einen wichtigen Schritt in Richtung Effizienz und Sicherheit darstellt.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ABKÜRZUNGSVERZEICHNIS

API Application Programming Interface

AWS Amazon Web Services

GCP Google Cloud Platform

RBAC Role-Based Access Control

DevOps Development and Operations

Azure Microsoft Azure

DNS Domain Name System

CAPI Cluster API

IaC Infrastructure as Code

API Application Programming Interface

TLD Top-Level Domain

VPC Virtual Private Cloud

DNSSEC Domain Name System Security Extensions

Teil I

ABSCHLUSSBERICHT

1 FIRMA

Die SDA SE ist ein in Hamburg ansässiges Unternehmen, das sich auf die Entwicklung von Plattformen und Ökosystemen spezialisiert hat.

1.1 FIRMENCHRONIK

Die Gründung der SDA SE erfolgte 2016 auf Initiative von Prof. Dr. Markus Warg, der als Ideengeber der SDA und Leiter des Instituts für Service Design in Hamburg, sowie Vorstand der SIGNAL IDUNA eine entscheidende Rolle spielte. Die SDA SE wurde als Joint Venture zwischen der SIGNAL IDUNA, der MSG-Systems AG, ALLIANZ X und der IBM gegründet.

Diese wegweisenden Kooperationen führten zur Entstehung der SDA SE.

Im letzten Jahr wurde der ehemalige CEO Dr. Stephan Hans gegangen, was bis heute einige weitere interims CEO's und Umstrukturierungen zur Folge hatte.

1.2 ZIELE UND PRODUKTE

Die SDA SE bietet eine digitale Service-Plattform mit einer einzigartigen serviceorientierten Architektur. Diese Plattform wird kontinuierlich von einem weltweiten Netzwerk führender Servicewissenschaftler weiterentwickelt.

Die Kernprinzipien sind Offenheit und Zusammenarbeit, die es Unternehmen ermöglichen, sich über eigene digitale Ökosysteme mit Partnern und Start-ups zu vernetzen. Alle Services basieren auf Open-Source-Software und sind darauf ausgerichtet, die Anforderungen der modernen digitalen Welt zu erfüllen.

1.3 BETRIEBSWIRTSCHAFTLICHE KENNZAHLEN

Leider können aktuelle betriebswirtschaftliche Kennzahlen aus vertraulichen Gründen nicht öffentlich zugänglich machen. Ehrlich gesagt sind diese auch persönlich gar nicht bekannt.

1.4 ORGANISATIONSSTRUKTUR

Die SDA SE ist ein Gemeinschaftsunternehmen von SIGNAL IDUNA, MSG, Allianz X, Debeka und HUK-COBURG. Im Aufsichtsrat finden sich Persönlichkeiten wie Prof. Dr. Markus Warg, Daniela Rode (Vorstand der SIGNAL IDUNA Gruppe), Carsten Middendorf (Head of Platforms, Acquisitions bei

Allianz X), Daniel Thomas (Vorstand der HUK-COBURG), Dr. Normann Pankratz (Vorstand der Debeka Versicherungsgruppe) sowie Dr. Jürgen Zehetmaier (Vorstand bei MSG-Systems).

Den heutigen Vorstand bildet Marco Ziegler.

1.5 PERSÖNLICHE AUFGABENSTELLUNG

Herr Siemer ist in seiner Position, als Development and Operations (DevOps) Engineer, für eine breite Palette von Aufgaben verantwortlich. Er entwickelt und implementiert eine Multicloud-Kubernetes-Plattform, die Amazon Web Services (AWS), Google Cloud Platform (GCP) und Microsoft Azure (Azure) umfasst. Diese Plattform wird von mehreren Teams und direkt von Kunden genutzt, um Microservices zu entwickeln und bereitzustellen.

Ziel ist es, eine effiziente Container-Orchestrierung und -Verwaltung über verschiedene Cloud-Plattformen sicherzustellen. Zusätzlich gestaltet und implementiert er eine umfassende Sicherheitsstrategie für die gesamte Infrastruktur.

Dabei werden spezielle Kubernetes-Sicherheitsmaßnahmen wie Netzwerkrichtlinien, Role-Based Access Control (RBAC) und Secrets Management eingesetzt. Er verantwortet auch das Management von Active Directory, die Autorisierung und Authentifizierung über verschiedene Flows von OAuth2. Herr Siemer arbeitet an der Entwicklung automatisierter Prozesse für Continuous Integration und Continuous Deployment und programmiert Cloud-native Software in den Sprachen Java, Python und Golang. Darüber hinaus ist er für die Erstellung von Microservice-Container-Images verantwortlich, um eine effiziente Bereitstellung und Skalierung zu ermöglichen.

Er implementiert Monitoring- und Alerting-Systeme und arbeitet an der Förderung einer Kultur der Zusammenarbeit und kontinuierlichen Verbesserung im Sinne von DevOps-Praktiken. Herr Siemer analysiert und behebt Infrastrukturprobleme und stellt die Systemverfügbarkeit und -leistung sicher. Erweitern des Cloud-Service-Angebots, die Optimierung von Cloud-Services und die Anpassung der Cloud-Infrastruktur an die Anforderungen des Unternehmens sind weitere Aufgaben.

2 PROJEKTBESCHREIBUNG

Der Titel des Projekts lautet „Automatische DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur“ und wird im Folgenden beschrieben. Diese Arbeit soll sich auf das Kernthema der automatischen DNS-Zonen und Eintragsverwaltung in einer verteilten Multi-Cloud-Microservice-Architektur konzentrieren und nicht auf die Implementierung der Microservices selbst. Domain Name System (DNS) bietet Namensauflösung für das Internet, indem es menschenlesbare Namen (z. B. example.com) in maschinenroutenfähige IP-Adressen (unter anderem) umwandelt. [Chu+17]. Weiterhin werden verwendete Technologien welche nicht direkt mit dem Kernthema in Verbindung stehen nur oberflächlich behandelt.

2.1 IDEE

Die Projektidee entstand aus dem Bedarf der Prozesstrennung und Automatisierung zwischen den Abteilungen der Entwicklung und dem Betrieb.

2.1.1 *Ziel*

Die Abteilung des Betriebs besteht aus DevOps, welche sich um die Entwicklung und Bereitstellung von Infrastruktur kümmert. Die Abteilungen der Entwickler beschäftigen sich mit der Entwicklung von Microservices und der Bereitstellung dieser auf der Infrastruktur. Microservices bezeichnen einen Ausbau einer Programm-Architektur, bei der ein Programm in mehrere kleine Programme aufgeteilt wird, welche jeweils eine Aufgabe erfüllen und mit anderen Teilen des Programms über Application Programming Interface (API)s kommunizieren [Lim23]. So ist ein Microservice eine kleine, unabhängige Komponente, welche eine Aufgabe erfüllt und mit anderen Microservices kommuniziert. Ein logischer Zusammenschluss aus mehreren Microservices bildet eine Anwendung, auch „service“ genannt.

Diese Infrastruktur wird auf Kundenwünsch hin auf verschiedenen Cloud-Providern bereitgestellt. Das Ziel der Arbeit ist den Entwicklern die Bereitstellung ihrer Microservices auf verschiedenen Cloud-Providern zu ermöglichen, ohne dass sie sich mit den Cloud-Providern beschäftigen müssen und ihnen eine adequate Abstraktionsebene zu bieten, welche die Komplexität der Cloud-Provider verbirgt und Prozesse außerhalb dieser Ebene automatisiert.

Fokus der Arbeit ist hierbei die automatisierung von DNS. DNS ist ein System, welches Domainnamen in IP-Adressen auflöst und somit die Erreich-

barkeit ermöglicht um Oberflächen oder APIs aufrufen zu können [Kleo3].

Weiterhin müssen alle Ergebnisse dieser Arbeit dem neuesten Stand der Technik entsprechen und die Sicherheit der Infrastruktur gewährleisten. So wurden vorab die Anforderungen an Private Zonen und DNSSEC ermittelt, welche im Verlauf der Arbeit genauer beschrieben werden. Private Zonen sind Zonen, welche nur innerhalb des Netzwerks erreichbar sind und somit die Sicherheit erhöhen, da IP-Adressen nicht öffentlich sichtbar sind. DNSSEC ist eine Erweiterung von DNS, welche die Authentizität und Integrität von DNS-Einträgen gewährleistet und somit die Sicherheit erhöht [Are+05].

2.1.2 *Motivation*

Für die Wahl der Abstraktionsebene gibt es viele Faktoren, welche nicht in gänze in dieser Arbeit behandelt werden können. Die Wahl fiel auf Kubernetes, da es sich um eine Open-Source Software handelt, welche Funktionen und Interfaces zur Verfügung stellt, um Microservices zu entwickeln und bereitzustellen. Kubernetes ist, nach Stand der Dinge, der Standard für Container Orchestration [Car22, p. 2]. Auf Kubernetes wird im Abschnitt 2.2.1 genauer eingegangen. Damit ein auf Kubernetes bereitgestellter Microservice von außerhalb des Kubernetes Clusters (z.B. aus dem Internet) erreichbar ist, muss ein DNS-Eintrag erstellt werden. Das sich nun stellende Problem besteht darin, dass die Microservices auf Kubernetes bereitgestellt werden, jedoch DNS-Einträge, welche für die Erreichbarkeit des Microservices sorgen, teil des Cloud-Providers sind und außerhalb von Kubernetes verwaltet werden. Dadurch entstand eine manuelle Tätigkeit, welche die Entwickler von ihrer eigentlichen Arbeit und Aufgabenbereich abhält und somit die Effizienz der Entwickler verringert. Schlussfolgernd ist für die SDA SE wünschenswert, den Zugriff der Entwickler so weit wie möglich auf lediglich Kubernetes zu beschränken und direkten Zugriff auf die Cloud-Provider zu vermeiden.

So entstand die Motivation, DNS-Einträge automatisch anhand der in Kubernetes enthaltenen Informationen zu erstellen und zu verwalten.

2.2 PLANUNG

Bei der Planung muss zwingend auf die Skalierbarkeit und Erweiterbarkeit der Lösung geachtet werden, da die SDA SE ein stark wachsendes Unternehmen ist und die Lösung auch in Zukunft noch verwendet werden soll. Außerdem muss die Lösung so gestaltet werden, dass sie auf verschiedenen Cloud-Providern funktioniert, da die SDA SE Microservices auf verschiedenen Cloud-Providern bereitstellt. Weiterhin ist das Lizenzmodell von verwendeten Lösungen zu beachten, da die SDA SE eine Open-Source first Policy verfolgt und somit Open-Source Lösungen bevorzugt werden.

2.2.1 *Umfeld*

Die Laufzeitumgebung der Microservices ist Kubernetes, welches auf den Cloud-Providern AWS, GCP und Azure als verwalteter Service gebucht wird. Kubernetes ist ein Container-Orchestrierungssystem, welches die Bereitstellung, Skalierung und Verwaltung von Containern ermöglicht. Die Microservices werden in den Programmiersprachen Java, Python und Golang entwickelt und in Docker Container-Images verpackt, was die enthaltenen Programmzeilen, Bibliotheken und Abhängigkeiten kapselt und somit eine effiziente Bereitstellung, Skalierung, Unveränderbarkeit und Verbreitung auf Kubernetes ermöglicht [RBA17]. Ein Docker Container-Image ist eine Datei, welche alle Abhängigkeiten und Konfigurationen enthält, um ein Programm zu starten und ist prinzipiell einer virtuellen Maschine sehr ähnlich nur, dass der Kernel des Hosts mitverwendet wird [RBA17, p. 229].

Die implementierung mehrerer Cloud-Provider ist notwendig, da die Microservices aufgrund von Kundenanforderungen auf verschiedenen Cloud-Providern bereitgestellt werden müssen. Ein Kunde kann beispielsweise die Bereitstellung seiner Microservices auf AWS fordern, während ein anderer Kunde die Bereitstellung auf GCP oder Azure fordert. Die Wahl des Cloud-Providers ist für Kunden eine essenzielle Entscheidung, da Kunden komplexe Verträge mit den Cloud-Providern abschließen und diese nicht ohne weiteres wechseln können.

Das Bereitstellen von Kubernetes auf verschiedenen Cloud-Providern ist eine komplexe Aufgabe, da die Cloud-Provider gänzlich unterschiedliche Herangehensweisen und Anforderungen haben. Der Aufbau der Infrastruktur auf AWS unterscheidet sich grundlegend von dem Aufbau auf GCP oder Azure, was einer vollständigen Automatisierung nicht gerade in die Hände spielt [Kho20, p. 450]. Neue Initiativen wie Cluster API (CAPI) versuchen diese Problematik zu lösen, benötigen jedoch auch viel Provider-spezifischen Code, was die Komplexität dadurch nicht verringert. Außerdem wird für diesen Ansatz ein management Kubernetes Cluster benötigt, welcher sich nicht selbst bootstrappen kann und somit zumindest initial ein anderer Ansatz gewählt werden muss. Bei der SDA SE wurde sich für eine Infrastructure as Code (IaC) Herangehensweise entschieden, welche die Infrastruktur auf den Cloud-Providern mittels Terraform bereitstellt. IaC ist ein Ansatz, bei dem die Infrastruktur mittels Code deklarativ beschrieben wird und somit eine automatisierte Bereitstellung ermöglicht. Terraform ist ein Open-Source Tool, welches die Infrastruktur auf verschiedenen Cloud-Providern bereitstellen kann, wohingegen ein konkurrierendes Tool wie CloudFormation nur mit AWS funktioniert. Hierbei verwendet Terraform die eigene Konfigurationssprache „HCL“, welche eine deklarative Beschreibung der Infrastruktur ermöglicht.

Trotz der Unterschiede bei den Cloud-Providern einen Kubernetes Cluster zu provisionieren, sind die Kubernetes Cluster implementationen bei den Cloud-Providern selbst beinahe identisch. Diesen Umstand will sich die

SDA SE zunutze machen und Kubernetes als Abstraktionsebene zwischen den Cloud-Providern und weiterer Infrastruktur nutzen, um die Komplexität vor den Entwicklern zu verbergen. Die Entwickler sollen sich nicht mit allen Feinheiten der Infrastruktur beschäftigen müssen, sondern lediglich Kubernetes nutzen, um ihre Microservices zu entwickeln und bereitzustellen, um ihre Arbeitszeit effizienter nutzen zu können. Um dies zu erreichen und die Entwickler so gut wie möglich mit automatischen Produktionsstrassen zu unterstützen, kann Kubernetes in seiner Funktion mit weiteren Programmen (auch Operatoren genannt) erweitert und ausgebaut werden. Ein solcher Ausbau wird häufig als Plattform bezeichnet.

So ist das Produkt der SDA SE eine Plattform (genannt „nimbusKube“), welche die Bereitstellung von Microservices auf verschiedenen Cloud-Providern mittels Kubernetes ermöglicht und erleichtert. Eine simultane Bereitstellung auf mehreren Cloud-Providern gleichzeitig ist möglich, jedoch im Standard nicht vorgesehen.

Eine Bestellung dieser Plattform enthält bis zu vier Kubernetes Clustern, welche auf dem gewählten Cloud-Provider bereitgestellt werden. Die Anzahl der Cluster ist abhängig von der Anzahl der gewünschten Umgebungen. So ist die kleinste Bestellung eine Umgebung mit zwei Clustern, einem „Tools“ und einem „Development“ Cluster. Der Tools Cluster ist für die Bereitstellung von Plattform-Tools zuständig, welche für die Entwicklung und Bereitstellung von Microservices benötigt werden. Die Umgebungscluster „Development“, „Staging“ und „Production“ sind einzig zuständig für das Bereitstellen von Firmen-Logik spezifischer Software.

2.2.2 *Plan*

Zur Projektplanung wurde ein iteratives Vorgehensmodell gewählt, da die Anforderungen und Auswirkungen des Projekts nicht vollständig bekannt sind und sich im Laufe des Projekts ändern können. Es wird außerdem angenommen, dass eine Konzeption sich später beim Test als unbefriedigend herausstellen könnte und somit ein Zurückkehren zur Konzeptionsphase notwendig sein könnte.

Phase	Tätigkeiten	Dauer (Stunden)
Anforderungen	Ziel definieren	3
Analyse	Vergleich Kubernetes Operatoren	5
	Funktionsweise DNSSEC	5
	Funktionsweise Public/Private	3
Konzeption	DNS Hierarchie	5
	Security Öffentlich/Private Zonen	5
	Security DNSSEC	5
	Infrastruktur Übersetzung	2
Implementierung	DNS Hierarchie	10
	Security Öffentlich/Private Zonen	15
	Security DNSSEC	10
	Infrastruktur Übersetzung	5
	Kubernetes Operator	10
Test	Testen der Erstellung	10
	Testen der Erreichbarkeit	3
Übergabe	Übergabe per IaC	0
Gesamt		100

Tabelle 2.1: Projektplan

2.2.3 *Wirtschaftlichkeitsbetrachtung*

Da zusätzliche Infrastrukturkosten derart gering sind, dass sie vernachlässigt werden können, wird sich die Wirtschaftlichkeitsbetrachtung auf die Arbeitszeit beschränken. So ist beispielhaft das Kostenmodell für DNS auf AWS nach Aufrufen gestaffelt und diese sind unabhängig vom Projekt gleich. Weiterhin sind die Kosten von Servern im Kubernetes auch unabhängig vom Projekt gleich, da ein Operator sehr geringe Lastkosten verursacht.

Die Projektlaufzeit wurde nach 2.1 auf 100 Stunden geschätzt. Die anfallende Arbeitszeit für die manuelle Erstellung DNS Einträge wurde auf 5 Minuten pro Eintrag geschätzt. Bei ungefähr 300 ausgerollten Microservices pro Tag (aus GitHub der SDA SE ausgelesen), ist der Break-Even-Point nach 4 Tagen erreicht.

2.2.4 *Alternativen*

Als alternative für den Ansatz welcher bereits in Teilen in dieser Arbeit präsentiert wurde, gibt es die Möglichkeit Wildcard DNS Einträge zu verwenden. Diese Einträge sind Platzhalter für alle Subdomains einer Domain und können auch zur Auflösung verwendet werden. Dieser Vorgang wäre nicht

auf die Erstellung einzelner Einträge angewiesen, sondern würde alle Subdomains einer Domain auflösen.

Dieser Ansatz erschwert jedoch die gewünschte Trennung in öffentliche und private Zonen, da sonst eine Subdomain zur Unterscheidung zwischen öffentlich und privat verwendet werden müsste. Auf die gewünschte Struktur der Zonen und Begründungen dieser wird im Abschnitt 2.3 genauer eingegangen.

2.3 UMSETZUNG

Diese Sektion befasst sich dem Verlauf der Realisierung des Projekts, der Organisation und den dabei aufgetretenen Problemen.

2.3.1 *Organisation*

Diese Arbeit wurde eigenständig und ohne direkte Anleitung durchgeführt. Die Organisation der Arbeit und der Arbeitsergebnisse wurde durch wöchentliche Meetings mit dem Auftraggeber sichergestellt.

Analysen und Konzepte wurden in Form von Dokumenten festgehalten und dem Auftraggeber zur Verfügung gestellt. Die Implementierung wurde in Form von Code in einem GitHub Repository festgehalten und dem Auftraggeber zur Verfügung gestellt.

2.3.2 *Verlauf*

Es wurde ein iteratives Modell gewählt, welche die Phasen Anforderungen, Analyse, Konzeption, Implementierung, Test immer wieder durchläuft. Eine Anforderung wurde definiert, der Ist-Zustand analysiert, ein Konzept erstellt um sich der Anforderung so weit wie möglich anzunähern, dieses Konzept implementiert und getestet. Daraufhin wurde die Anforderung erneut betrachtet und gegebenenfalls angepasst. Beim zufriedenstellenden Ergebnis wurde die nächste Anforderung betrachtet und der Prozess wiederholt. Mit dem Abschließen der letzten Anforderung, wurden die Ergebnisse in ihrer Gänze betrachtet, getestet und gegebenenfalls angepasst. Am Ende der Arbeit wurde die Übergabe der Ergebnisse an den Auftraggeber durchgeführt.

2.4 ERGEBNISSE

Diese Sektion befasst sich mit den Ergebnissen der Arbeit und der Wirtschaftlichkeitsbetrachtung.

2.4.1 *Darstellung*

Das erste zu präsentierende Ergebnis ist die Auswahl des Kubernetes Operators. Hierbei wurde sich für den Operator „external-dns“ entschieden, da

dieser die gewünschten Funktionen bietet und die SDA SE bereits Erfahrung mit diesem Operator hat. Dieser Operator ist aktuell der einzige Open-Source Operator, welcher die gewünschten Funktionen bietet.

Als nächstes soll die DNS Hierarchie dargestellt werden. Die Domain ist „sda-se.io“, wobei die Top-Level Domain (TLD) (io) bei einem externen Registrar registriert ist. Registrare sind Organisationen, die Domainnamen verkaufen und häufig die autoritativen Nameserver betreiben [Chu+17].

In einem AWS Konto, beispielsweise mit dem Namen „DNS-Root“, werden nun eine private (private) und eine öffentliche (public) Zone für „sda-se.io“ erstellt. Die Nameserver der öffentlichen Domain werden beim Registrar eingetragen, um zur öffentlichen Zone zu delegieren.

Wird eine Bestellung der Plattform getätigt, werden in einem beliebigen Cloud-Provider (z.B. AWS) weitere Zonen erstellt. Die Bestellung der Plattform erhält einen Projektnamen, welcher als Subdomain der Domain verwendet wird, zum Beispiel „projekt1“. Für diesen Namen werden in dem Zielkonto eine öffentliche und eine private Zone erstellt und die private Zone mit dem Virtual Private Cloud (VPC) der Plattform assoziiert. Mit einem AWS VPC kann eine logisch isolierte Sektion der AWS-Cloud bereitgestellt werden, in der AWS-Ressourcen in einem definierten virtuellen Netzwerk gestartet werden können [MV14, p. 12]. Über diese Assoziation wird implizit „DNS Split-Horizon“ konfiguriert. Split-horizon DNS, auch als Split-View DNS, Split-Brain DNS oder Split DNS bekannt, ermöglicht es, verschiedene Sätze von DNS-Informationen bereitzustellen, die in der Regel durch die Quelladresse der DNS-Anfrage ausgewählt werden. Diese Funktion bietet einen Mechanismus für das Management von Sicherheit und Privatsphäre durch die logische oder physische Trennung von DNS-Informationen für den Netzwerk internen Zugriff [con23].

Weiterhin wird für jede Umgebung, die dazu bestellt wurde, ein weiterer Satz Zonen erstellt (privat und öffentlich), welche die Subdomain der Umgebung enthält (außer Produktion, diese Umgebung verwendet die Projekt-Subdomain). So werden für die Umgebung „Development“ die Zonen für „dev.projekt1.sda-se.io“ erstellt.

Für die Veröffentlichung eines Services wird eine „VirtualService“ Resource in Kubernetes erstellt, welches die gewünschte Subdomain enthält. Ein VirtualService ist eine Kubernetes Resource, welche von dem Istio Service Mesh verwendet wird, um die Veröffentlichung von Services zu ermöglichen. Istio ist ein Service-Mesh, dass die Netzwerkkommunikation zwischen Mikrodiensten in einer Kubernetes-Umgebung verwaltet, überwacht und sichert.

In dem VirtualService wird die gewünschte Subdomain des Services angegeben, welche vom Operator „external-dns“ beobachtet werden muss. Außerdem muss ein Istio Ingress Gateway angegeben werden (privat oder öffentlich). Der Operator „external-dns“ beobachtet die VirtualService Resource und erstellt automatisch einen DNS-Eintrag in der entsprechenden Zone.

Nachdem ein grundlegendes Verständnis für diesen Prozess geschaffen wurde, kann nun die Sicherheit der Zonen betrachtet werden. Öffentliche Zonen sollten nur DNS-Einträge öffentlicher Services enthalten, bei denen die veröffentlichung der Einträge unabdingbar ist. DNS-Einträge privater Services sollten nur in privaten Zonen enthalten sein, welche nur innerhalb des Netzwerks erreichbar sind, damit eventuelle Angreifer keine Informationen über die Infrastruktur erhalten können. Dies wird mittels Split-Horizon DNS und der Konfiguration von external-dns erreicht. Jede Zone erhält eine Installation des Operators external-dns. Die Installation der öffentlichen Zone erstellt nur die DNS-Einträge der öffentlichen Services in der öffentlichen Zone. Die Installation der privaten Zone erstellt alle DNS-Einträge, jene der privaten als auch die der öffentlichen Services in der privaten Zone. Anfragen, welche aus dem Internet erfolgen, werden an die öffentliche Zone delegiert und erhalten nur die DNS-Einträge der öffentlichen Services. Anfragen, welche aus dem privaten Netzwerk erfolgen, werden an die private Zone delegiert (Split-Horizon) und erhalten alle DNS-Einträge, jene der privaten als auch die der öffentlichen Services.

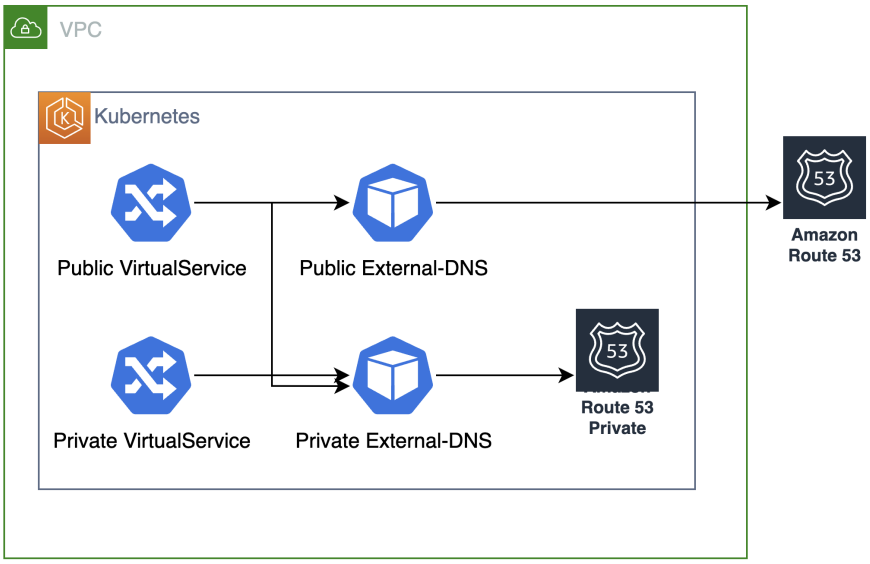

Abbildung 2.1: Private und öffentliche Zonenverwaltung mit external-dns

Ein in Kubernetes veröffentlichter Service, zum Beispiel „service1“, wird in der VirtualService Resource mit der Subdomain „service1.dev.projekt1.sdase.io“ angegeben und als öffentlicher Service konfiguriert in dem es an das öffentliche Istio Ingress Gateway gebunden wird. So wird der Service mit seiner Subdomain in sowohl der öffentlichen als auch der privaten Zone erstellt.

Die Sicherheit der Zonen wird durch die Verwendung von Domain Name System Security Extensions (DNSSEC) gewährleistet. DNSSEC ist eine Erweiterung von DNS, welche die Authentizität und Integrität von DNS-Einträgen gewährleistet und somit die Sicherheit erhöht [Kelo1]. Um bei

AWS DNSSEC zu aktivieren, muss ein KMS-Schlüssel erstellt werden, welcher für die Signierung der DNS-Einträge verwendet wird. Dieser Schlüssel wird in der Route53 Zone angegeben und aktiviert DNSSEC für diese Zone. Außerdem muss eine Kette des Vertrauens erstellt werden, welche durch „DS“ Records in der jeweiligen Übergeordneten Zone hinterlegt werden. Nachfolgend wird die Verwaltung der öffentlichen Zonen beispielhaft mit Terraform dargestellt. Es handelt sich um ein vereinfachtes Beispiel, welches die wesentlichen Teile der Konfiguration darstellt aber nicht so komplex ist, wie die tatsächliche Konfiguration.

Listing 2.1: DNSSEC mit Terraform

```
provider "aws" {
  region = "eu-central-1"
}

data "aws_caller_identity" "current" {}

resource "aws_kms_key" "dnssec_key" {
  customer_master_key_spec = "ECC_NIST_P256"
  deletion_window_in_days  = 7
  key_usage                = "SIGN_VERIFY"
  policy = jsonencode({
    Statement = [
      {
        Action = [
          "kms:DescribeKey",
          "kms:GetPublicKey",
          "kms:Sign",
        ],
        Effect = "Allow",
        Principal = {
          Service = "dnssec-route53.amazonaws.com"
        },
        Sid      = "Allow Route 53 DNSSEC Service",
        Resource = "*",
        Condition = {
          StringEquals = {
            "aws:SourceAccount" = data.aws_caller_identity.current.
                account_id
          },
          ArnLike = {
            "aws:SourceArn" = "arn:aws:route53:::hostedzone/*"
          }
        }
      },
      {
        Action = "kms:CreateGrant",
        Effect = "Allow",
        Principal = {
          Service = "dnssec-route53.amazonaws.com"
        },
```

```
        Sid      = "Allow Route 53 DNSSEC Service to CreateGrant",
        Resource = "*",
        Condition = {
          Bool = {
            "kms:GrantIsForAWSResource" = "true"
          }
        }
      },
      {
        Action = "kms:*",
        Effect = "Allow",
        Principal = {
          AWS = "arn:aws:iam::${data.aws_caller_identity.current.
              account_id}:root"
        },
        Resource = "*",
        Sid      = "Enable IAM User Permissions"
      },
    ],
    Version = "2012-10-17"
  })
}

resource "aws_route53_zone" "sda_se_io_zone" {
  name = "sda-se.io"
}

resource "aws_route53_zone" "projekt1_public_zone" {
  name = "projekt1.sda-se.io"
}

resource "aws_route53_zone" "dev_projekt1_public_zone" {
  name = "dev.projekt1.sda-se.io"
}

resource "aws_route53_record" "sda_se_io_to_projekt1" {
  zone_id = aws_route53_zone.sda_se_io_zone.zone_id
  name    = "projekt1"
  type    = "NS"
  ttl     = 300
  records = aws_route53_zone.projekt1_public_zone.name_servers
}

resource "aws_route53_record" "projekt1_to_dev_projekt1_public" {
  zone_id = aws_route53_zone.projekt1_public_zone.zone_id
  name    = "dev"
  type    = "NS"
  ttl     = 300
  records = aws_route53_zone.dev_projekt1_public_zone.name_servers
}

resource "aws_route53_key_signing_key" "dev_projekt1_public_zone" {
```

```
  hosted_zone_id = aws_route53_zone.dev_projekt1_public_zone.id
  key_management_service_arn = aws_kms_key.dnssec_key.arn
  name                       = "dev"
}

resource "aws_route53_hosted_zone_dnssec" "dev_projekt1_public_zone" {
  hosted_zone_id = aws_route53_zone.dev_projekt1_public_zone.id
}

resource "aws_route53_key_signing_key" "projekt1_public_zone" {
  hosted_zone_id = aws_route53_zone.projekt1_public_zone.id
  key_management_service_arn = aws_kms_key.dnssec_key.arn
  name                       = "projekt1"
}

resource "aws_route53_hosted_zone_dnssec" "projekt1_public_zone" {
  hosted_zone_id = aws_route53_zone.projekt1_public_zone.id
}

resource "aws_route53_record" "projekt1_to_dev_projekt1_public" {
  zone_id = aws_route53_zone.projekt1_public_zone.zone_id
  name    = "dev"
  type    = "DS"
  ttl     = 300
  records = [
    aws_route53_key_signing_key.dev_projekt1_public_zone.ds_record,
  ]
}

resource "aws_route53_key_signing_key" "projekt1_public_zone" {
  hosted_zone_id     = aws_route53_zone.sda_se_io_zone.id
  key_management_service_arn = aws_kms_key.dnssec_key.arn
  name                       = "sda-se"
}

resource "aws_route53_hosted_zone_dnssec" "projekt1_public_zone" {
  hosted_zone_id = aws_route53_zone.sda_se_io_zone.id
}

resource "aws_route53_record" "projekt1_to_dev_projekt1_public" {
  zone_id = aws_route53_zone.sda_se_io_zone.zone_id
  name    = "projekt1"
  type    = "DS"
  ttl     = 300
  records = [
    aws_route53_key_signing_key.projekt1_public_zone.ds_record,
  ]
}
```

Mittels Terraform kann dieser Code-Ausschnitt geplant und ausgeführt werden, um die enthaltenen Ressourcen zu erstellen. Nach erfolgreicher Aus-

führung kann die Konfiguration mittels verschiedenen Linux Tools wie „dig“ oder „nslookup“ geprüft und getestet werden.

2.4.2 *Analyse*

Der nach der Darstellung herbeigeführte Zustand entspricht im vollsten Umfang den Anforderungen und wurde erfolgreich getestet. Die Anforderungen wurden in ihrer Gänze erfüllt und die Ergebnisse entsprechen dem Stand der Technik. Folgende Anforderungen wurden erfüllt:

- Es wurde ein Ziel definiert und erreicht.
- Kubernetes DNS Operatoren wurden vergleicht, ausgewählt und implementiert.
- Ein DNS-Hierarchie-Konzept wurde erstellt und implementiert.
- Ein Konzept für private und öffenliche Zonen wurde erstellt und implementiert.
- Ein Konzept für DNSSEC wurde erstellt und implementiert.
- Eine Infrastruktur-Übersetzung wurde mittels Terraform erstellt und implementiert.
- Die Ergebnisse wurden getestet und entsprechen dem Stand der Technik.

2.4.3 *Wirtschaftlichkeitsbetrachtung*

Die Wirtschaftlichkeitsbetrachtung wurde wie in Abschnitt 2.2.3 geplant durchgeführt und der Break-Even Point nach 4 Tagen erreicht. Die Wirtschaftlichkeit der Lösung ist somit gegeben und die Lösung kann in Produktion gehen.

2.5 SCHLUSSFOLGERUNG

Diese Sektion befasst sich mit den Auswirkungen, Modifikationen und der Bilanz der Arbeit.

2.5.1 *Auswirkungen*

Die Auswirkungen der Arbeit sind, dass die Entwickler der SDA SE nun ihre Microservices auf verschiedenen Cloud-Providern bereitstellen können, ohne sich mit der DNS Konfiguration der Cloud-Provider beschäftigen zu müssen. Dies ermöglicht den Entwicklern sich auf ihre eigentliche Arbeit zu konzentrieren und die Effizienz der Entwickler zu erhöhen. Es hat viel Zuspruch von den Entwicklern gegeben und die Arbeit wurde als sehr hilfreich empfunden.

2.5.2 *Modifikationen*

Das Ziel der Arbeit wurde erreicht und die Anforderungen wurden erfüllt. Die Arbeit kann in Produktion gehen und die Entwickler können die Ergebnisse der Arbeit verwenden. Es waren bisher keine Modifikationen notwendig und sind auch nicht geplant.

2.5.3 *Bilanz*

Die Entwickler können sich nun auf ihre eigentliche Arbeit konzentrieren und sind nicht mehr mit manuellen Tätigkeiten beschäftigt. Der Plan, die Entwickler so weit wie möglich von der DNS Infrastruktur zu entkoppeln, wurde erfolgreich umgesetzt und die Erstellung und Verwaltung der SDA SE DNS Systeme automatisiert. Gerade in der heutigen Zeit, in der die Automatisierung von Prozessen immer wichtiger wird, ist dies ein großer Schritt in die richtige Richtung. Gelerntes Wissen über DNS und Kubernetes kann in zukünftigen Projekten verwendet werden und die SDA SE kann sich auf die Ergebnisse der Arbeit verlassen. Weiterhin wurde die Sicherheit der Infrastruktur erhöht, da die DNS-Einträge nun mittels DNSSEC signiert werden und die Zonen in öffentliche und private Zonen aufgeteilt wurden. Dies ist ein großer Schritt in Richtung einer sicheren Infrastruktur und erhöht die Sicherheit der SDA SE und wird dem Unternehmen in Zukunft zugute kommen.

LITERATUR

[Are+05] Roy Arends, Rob Austein, Matt Larson, Dan Massey und Scott Rose. *RFC 4033: DNS security introduction and requirements*. 2005.

[Car22] Carmen Carrión. „Kubernetes as a Standard Container Orchestrator - A Bibliometric Analysis“. In: *Journal of Grid Computing* 20.4 (2022), S. 42. DOI: 10.1007/s10723-022-09629-8. URL: https://doi.org/10.1007/s10723-022-09629-8.

[Chu+17] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove und Christo Wilson. „Understanding the role of registrars in DNSSEC deployment“. In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, S. 369–383.

[Kel01] Stefan Kelm. „Zur Sicherheit von DNS (DNSSEC)“. In: *Verlässliche IT-Systeme 2001: Sicherheit in komplexen IT-Infrastrukturen* (2001), S. 107–124.

[Kho20] Aditi Rajan Khot. „A Comparative Analysis of Public Cloud Platforms and Introduction of Multi-Cloud“. In: *International Journal of Innovative Science and Research Technology* 5.9 (2020), S. 448–454.

[Kle03] John Klensin. *Role of the domain name system (dns)*. Techn. Ber. 2003.

[Lim23] Red Hat Limited. *Was sind Microservices?* https://www.redhat.com/de/topics/microservices/what-are-microservices. [Online; accessed 04-November-2023]. 2023.

[MV14] Sajee Mathew und J Varia. „Overview of amazon web services“. In: *Amazon Whitepapers* 105 (2014), S. 1–22.

[RBA17] Babak Bashari Rad, Harrison John Bhatti und Mohammad Ahmadi. „An introduction to docker and analysis of its performance“. In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), S. 228.

[con23] Wikipedia contributors. *Split-horizon DNS*. https://en.wikipedia.org/wiki/Split-horizon_DNS. [Online; accessed 25-November-2023]. 2023.