

# Projektabgabe: DevOps

Hammerschmidt, Rentenberger, Schodl, Weidinger

21. Januar 2025

## Inhaltsverzeichnis

<b>1</b>	<b>Netzwerktopologie</b>	<b>4</b>
<b>2</b>	<b>Geplante Security Groups und Regeln</b>	<b>5</b>
<b>3</b>	<b>Spezifikationen der eingesetzten Systeme</b>	<b>6</b>
<b>4</b>	<b>Tests</b>	<b>7</b>
<b>5</b>	<b>Rollen und Verantwortlichkeiten im Team</b>	<b>10</b>
<b>6</b>	<b>Monitoring</b>	<b>10</b>
<b>7</b>	<b>VPC erstellen</b>	<b>10</b>
<b>8</b>	<b>Creating Subnets</b>	<b>11</b>
8.1	Public Subnet . . . . .	11
8.2	Private Subnet . . . . .	11
8.2.1	Enabling Auto-assign IP for Public Subnet . . . . .	11
<b>9</b>	<b>Internet Access</b>	<b>12</b>
9.1	Enabling Internet Access for Public Subnet . . . . .	12
9.1.1	Internet Gateway . . . . .	12
9.1.2	Routing Table . . . . .	13
9.2	Enabling Internet Access for Private Subnet . . . . .	13
9.2.1	NAT Gateway . . . . .	13
<b>10</b>	<b>Security Groups</b>	<b>15</b>
10.1	Public GitLab . . . . .	15
10.2	Private GitLab . . . . .	15
10.3	DNS . . . . .	15
10.4	LDAP . . . . .	16
<b>11</b>	<b>Launch Instance</b>	<b>16</b>

<b>12 SSH Access zu Server im Private Subnet</b>	<b>20</b>
12.1 Setting up SSH Agent Forwarding . . . . .	20
12.2 SSH Zugriff . . . . .	21
12.3 Aktualisieren des Betriebssystems . . . . .	21
<b>13 Setup der DNS Server</b>	<b>21</b>
13.1 Installation von BIND9 . . . . .	21
13.2 Konfiguration des Primary DNS Servers . . . . .	21
13.2.1 Erstellung der Access Contol List . . . . .	22
13.2.2 Konfiguration der Allgemeine Optionen . . . . .	22
13.2.3 Konfiguration des “Local” Files . . . . .	22
13.2.4 Hinzufügen der Forward Zone . . . . .	22
13.2.5 Hinzufügen der Reverse Zone für das Public Subnet . . . . .	23
13.2.6 Hinzufügen der Reverse Zone für das Private Subnet . . . . .	23
13.2.7 Erstellen des Forward Zone Files . . . . .	23
13.2.8 Erstellen des Reverse Zone Files für das Public Subnet . . . . .	24
13.2.9 Erstellen des Reverse Zone Files für das Private Subnet . . . . .	26
<b>14 Konfiguration des sekundären DNS-Servers</b>	<b>26</b>
14.1 Einrichten der Zugriffskontrollliste und allgemeiner Optionen . . . . .	27
14.2 Konfigurieren vom Local File . . . . .	27
<b>15 Konfiguration der Server</b>	<b>28</b>
15.1 Festlegen der Nameserver . . . . .	28
15.2 Änderungen dauerhaft speichern . . . . .	28
15.3 Testen der Nameserver . . . . .	29
15.4 Überprüfen, ob der sekundäre DNS-Server wie vorgesehen funktioniert . .	29
<b>16 GitLab</b>	<b>30</b>
16.1 Konfiguration des SSH-Ports . . . . .	30
16.2 Anpassen der Sicherheitsgruppe auf AWS . . . . .	30
16.3 Installation von GitLab CE mit Docker Compose . . . . .	30
16.3.1 Erstellen von Verzeichnissen zur Datenpersistenz . . . . .	30
16.3.2 Erstellen des Containers mit Docker Compose . . . . .	31
<b>17 GitLab Runner</b>	<b>33</b>
17.1 Installation des GitLab Runners mit Docker . . . . .	33
17.2 Registrierung des Runners in GitLab . . . . .	33
17.2.1 Erstellen des Runner-Authentifizierungstokens . . . . .	33
17.2.2 Registrierung des Runners . . . . .	33
<b>18 LDAP Server</b>	<b>34</b>
18.1 Vorbereitung . . . . .	34
18.1.1 Instance erstellen . . . . .	35
18.1.2 Security Group . . . . .	35
18.1.3 DNS Eintrag für LDAP Server . . . . .	35
18.1.4 Zone Files . . . . .	35
18.2 LDAP . . . . .	35
18.2.1 Verzeichnis füllen . . . . .	36

18.2.2 LDAP-Server Verbindung testen . . . . .	37
18.3 Integration von LDAP mit GitLab . . . . .	37
<b>Abbildungsverzeichnis</b>	<b>39</b>
<b>Tabellenverzeichnis</b>	<b>40</b>

# 1 Netzwerktopologie

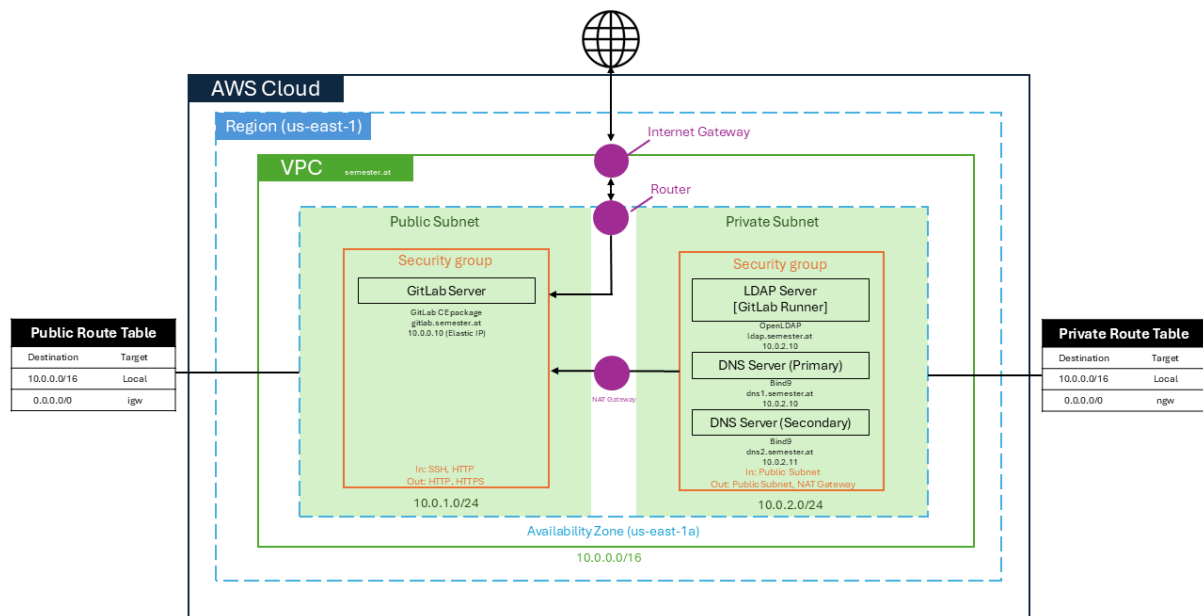


Abbildung 1: Netzwerktopologie der Infrastruktur

Dienst	Subnetztyp	IP-Adresse	FQDN
Primärer DNS	Privates Subnetz	10.0.2.225	ns1.semesterDevOps.com
Sekundärer DNS	Privates Subnetz	10.0.2.10	ns2.semesterDevOps.com
GitLab Server	Öffentliches Subnetz	Public	gitlab.semesterDevOps.com
LDAP Server	Privates Subnetz	10.0.2.10	ldap.semesterDevOps.com
GitLab-Runner	Privates Subnetz	10.0.2.184	gitlab.semesterDevOps.com

Tabelle 1: Netzwerkdienste und IP-Zuordnung

## 2 Geplante Security Groups und Regeln

<b>Inbound SGRs</b>	<b>DNS</b>	<b>LDAP/GitLab-R</b>	<b>GitLab-Server</b>
HTTP	Nein	Nein	Ja
HTTPS	Nein	Ja	Ja
SSH	Ja	Ja	Ja
DNS (UDP)	Ja	Nein	Nein
DNS (TCP)	Ja	Nein	Nein
LDAP	Nein	Ja	Ja
ALL ICMP	Ja	Nein	Ja

Tabelle 2: Eingehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste

<b>Outbound SGRs</b>	<b>DNS</b>	<b>LDAP/GitLab-R</b>	<b>GitLab-Server</b>
HTTP	Nein	Ja	Nein
HTTPS	Nein	Ja	Nein
SSH	Nein	Ja	Nein
DNS (UDP)	Nein	Ja	Nein
DNS (TCP)	Nein	Ja	Nein
LDAP	Nein	Ja	Ja
ALL ICMP	Nein	Ja	Nein
ALL Traffic	Ja	Nein	Ja

Tabelle 3: Ausgehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste

### 3 Spezifikationen der eingesetzten Systeme

Server	OS	Packages	Version	Server Instance
Primärer DNS Server	Ubuntu Server	bind9, bind9utils	BIND 9 / Ubuntu 24.04 LTS	T3.micro
Sekundärer DNS Server	Ubuntu Server	bind9, bind9utils	BIND 9 / Ubuntu 24.04 LTS	T3.micro
LDAP Server	Ubuntu Server	slapd, ldap-utils	OpenLDAP 2.6	T3.micro
GitLab Runner	Ubuntu Server	-	Ubuntu 24.04 LTS	T3.micro
GitLab Server	Ubuntu Server	GitLab CE	GitLab CE / Ubuntu 24.04 LTS	T2.Large

Tabelle 4: Server-Spezifikationen: Betriebssystem, Pakete und Instanztypen

- **Betriebssystem:** Ubuntu 24.04 LTS LTS (64-bit)
- **DNS-Server:** BIND 9.x
- **GitLab:** GitLab CE 15.x
- **GitLab Runner:** Version kompatibel mit GitLab CE 15.x
- **LDAP:** OpenLDAP 2.6.x
- **Monitoring:** AWS CloudWatch zur Protokollierung und Überwachung.

## 4 Tests

### DNS Resolution Testing

- **Ziel:** Sicherstellen, dass der BIND-Server Domain-Namen korrekt auflöst.
- **Methode:** Verwenden des `dig`-Befehls, um den DNS-Server nach bekannten Domains abzufragen. Überprüfen der A, AAAA, MX und NS Records:

```
dig @<DNS-server> example.com [A, AAAA, MX, NS]
```

- **Erwartetes Ergebnis:** Jede Abfrage liefert die richtigen IP-Adressen und Record-Details.

### Forward and Reverse DNS Lookup

- **Ziel:** Überprüfen, dass Vorwärts- und Rückwärts-Abfragen funktionieren.
- **Methode:**

- Verwenden von `dig` für die Vorwärtsabfrage (Domain zu IP):

```
dig @<DNS-server> example.com A
```

- Verwenden von `dig -x` für die Rückwärtsabfrage (IP zu Domain):

```
dig @<DNS-server> -x [192.0.2.1]
```

- **Erwartetes Ergebnis:** Genaues Mapping zwischen Domain-Namen und IP-Adressen.

### Zone Transfer Test

- **Ziel:** Sicherstellen, dass Zonentransfers zwischen primären und sekundären DNS-Servern funktionieren.
- **Methode:** Einen Zonentransfer mit `dig AXFR` anstoßen und die Logs auf den Transfer überprüfen:

```
dig @<primary-DNS-server> example.com AXFR
```

- **Erwartetes Ergebnis:** Zonendaten werden korrekt zwischen primären und sekundären Servern repliziert.

## DNS Failover Testing

- **Ziel:** Die Resilienz und Zuverlässigkeit des DNS-Dienstes unter Ausfallbedingungen bewerten.

- **Methode:**

- Einen Ausfall des primären DNS-Servers simulieren.
- Die Antwort des sekundären DNS-Servers überwachen:

```
dig @<secondary-DNS-server> example.com A
```

- Etwaige Ausfallzeiten während des Übergangs protokollieren.

- **Erwartetes Ergebnis:** Der sekundäre DNS-Server übernimmt nahtlos mit wenig bis gar keiner Unterbrechung der DNS-Auflösung.

## Stress Testing

- **Ziel:** Die Leistung des Servers unter hoher Last testen.
- **Methode:** Tools wie `dnstperf` verwenden, um eine hohe Anzahl von DNS-Abfragen zu simulieren:

```
dnstperf -s <primary-DNS-IP> -d queries.txt -l 30
```

- **Erwartetes Ergebnis:** Der Server bleibt auch unter Last genau und leistungsfähig.

## LDAP-Authentifizierungstest

### LDAP-Authentifizierungstest

- **Ziel:** Funktionalität des Authentifizierungssystems überprüfen.
- **Methode:** Benutzeranmeldungen über LDAP versuchen. Tests mit gültigen und ungültigen Anmeldedaten durchführen.
- **Erwartetes Ergebnis:** Anmeldeversuche sollten für gültige Anmeldedaten akzeptiert und für ungültige Anmeldedaten abgelehnt werden.

## LDAP-Suche und -Filterung

- **Ziel:** Genauigkeit der LDAP-Suche und -Filterung bestätigen.
- **Methode:** Suchen nach bestimmten Benutzergruppen oder Attributen durchführen und Filter testen.
- **Erwartetes Ergebnis:** Für jede Suche und jeden Filter werden die korrekten Daten zurückgegeben.



## Benutzer- und Gruppenverwaltung

- **Ziel:** Testen der Erstellung und Änderung von Benutzern und Gruppen.
- **Methode:** Gruppen und Benutzer erstellen sowie deren Attribute ändern. Änderungen im LDAP-Verzeichnis überwachen.
- **Erwartetes Ergebnis:** Alle Änderungen werden korrekt im LDAP-Verzeichnis angezeigt.

## LDAP-Integrationstest

- **Ziel:** Testen, ob die Integration der GitLab-Authentifizierung mit LDAP funktioniert.
- **Methode:** Anmeldungen bei GitLab mit LDAP-Anmeldedaten (verschiedene Rollen) durchführen.
- **Erwartetes Ergebnis:** Benutzeranmeldungen werden mit LDAP-Anmeldedaten akzeptiert.

## Repository-Operationen

- **Ziel:** Testen, ob Standard-Git-Operationen innerhalb von GitLab funktionieren.
- **Methode:** Neben der Erstellung von Repositories werden Pull-, Push- und Merge-Operationen getestet. Zusätzlich werden Branches erstellt und gelöscht.
- **Erwartetes Ergebnis:** Änderungen entsprechen den erwarteten Ergebnissen der Git-Operationen.

## CI/CD-Pipeline-Test

- **Ziel:** Funktionalität der CI/CD-Pipeline überprüfen.
- **Methode:** Geänderten Code pushen und die automatische Ausführung der CI/CD-Pipeline beobachten. Erfolg des Builds und der Bereitstellung prüfen.
- **Erwartetes Ergebnis:** Codeänderungen werden automatisch gebaut und fehlerfrei bereitgestellt.

## Lasttest

- **Ziel:** Leistung von GitLab unter hoher Last bewerten.
- **Methode:** Mehrere Benutzer simulieren, die gleichzeitig Git-Operationen durchführen.
- **Erwartetes Ergebnis:** GitLab hält die Leistungsniveaus auch bei gleichzeitiger Nutzung aufrecht.

## 5 Rollen und Verantwortlichkeiten im Team

Samuel Hammerschmidt	Lorenz Rentenberger	Nikolas Schodl	Alexander Weidinger
LDAP Server	DNS Server (CloudWatch)	GitLab Server	GitLab Server
AWS Cloud Config	AWS Cloud Config	AWS Cloud Config	AWS Cloud Config
Tests LDAP	Tests DNS	Tests GitLab	Tests GitLab

Tabelle 5: Team-Aufgaben und Zuständigkeiten

## 6 Monitoring

AWS CloudWatch wird zur Protokollierung und Überwachung genutzt:

- Überwachung der CPU-, Speicher- und Netzwerknutzung.
- Automatische Alarmer bei Ausfällen.

## 7 VPC erstellen

Um ein VPC zu erstellen müssen wir zuerst sicherstellen, dass wir in der richtigen Region sind. In unserem Fall ist das 'us-east-1'.

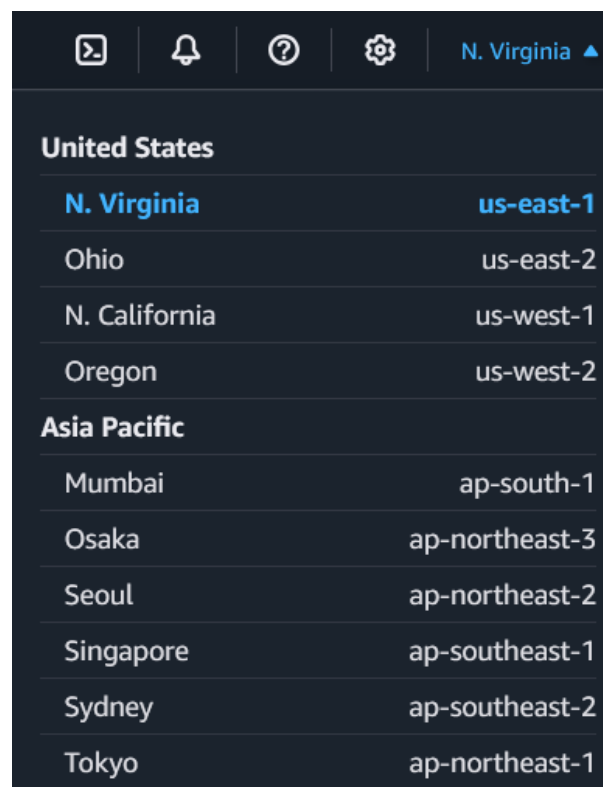


Abbildung 2: Auswahl der Region

## 8 Creating Subnets

### 8.1 Public Subnet

Zuerst erstellen wir ein öffentliches Subnet. Wie im Bild unten angeführt, wählen wir den richtigen VPC(10.0.0.0/16) aus und konfigurieren den privaten CIDR-Block(10.0.1.0/24).

**Create subnet** [Info](#)

**VPC**  
VPC ID  
Create subnets in this VPC.  
vpc-0074580bc28d455f5 (semester-vpc)

**Associated VPC CIDRs**  
IPv4 CIDRs  
10.0.0.0/16

**Subnet settings**  
Specify the CIDR blocks and Availability Zone for the subnet.

**Subnet 1 of 1**

**Subnet name**  
Create a tag with a key of 'Name' and a value that you specify.  
semester-public-sn  
The name can be up to 256 characters long.

**Availability Zone** [Info](#)  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.  
No preference

**IPv4 VPC CIDR block** [Info](#)  
Choose the VPC's IPv4 CIDR block for the subnet. The subnet's IPv4 CIDR must lie within this block.  
10.0.0.0/16

**IPv4 subnet CIDR block**  
10.0.1.0/24 256 IPs

**Tags - optional**

Key	Value - optional	
Q Name	Q semester-public-sn	Remove
Q Environment	Q Semester	Remove

[Add new tag](#)  
You can add 48 more tags.

Abbildung 3: Erstellung des Public Subnet

### 8.2 Private Subnet

Beim Erstellen vom Public Subnet, gehen wir die selbe Schritte durch, ändern den CIDR-Block jedoch auf 10.0.2.0/24.

#### 8.2.1 Enabling Auto-assign IP for Public Subnet

Wichtig zum Erwähnen ist die Aktivierung der Option 'Enable auto-assign public IPv4 address'. Das sorgt dafür, dass jeder neu erstellten EC2 Instanz eine neue IP Adresse zugewiesen wird.

**Edit subnet settings** [Info](#)

**Subnet**  
Subnet ID  
 subnet-08ad7172adeb52985

**Name**  
 semester-public-sn

**Auto-assign IP settings** [Info](#)  
Enable AWS to automatically assign a public IPv4 or IPv6 address to a new primary network interface for an instance in this subnet.

☒ Enable auto-assign public IPv4 address [Info](#)

☐ Enable auto-assign customer-owned IPv4 address [Info](#)  
Option disabled because no customer owned pools found.

**Resource-based name (RBN) settings** [Info](#)  
Specify the hostname type for EC2 instances in this subnet and optional RBN DNS query settings.

☐ Enable resource name DNS A record on launch [Info](#)

☐ Enable resource name DNS AAAA record on launch [Info](#)

**Hostname type** [Info](#)  
☐ Resource name  
☒ IP name

**DNS64 settings**  
Enable DNS64 to allow IPv6-only services in Amazon VPC to communicate with IPv4-only services and networks.

☐ Enable DNS64 [Info](#)

[Cancel](#) [Save](#)

Abbildung 4: Auto-Assign aktivieren

## 9 Internet Access

### 9.1 Enabling Internet Access for Public Subnet

Weiters muss das Public Subnet noch Zugriff auf das Internet bekommen. Dafür verwenden wir Route-Tables, die wiederum Zugriff auf das Internet Gateway gewähren.

#### 9.1.1 Internet Gateway

Zuerst erstellen wir ein Gateway, mit dem Namen 'semester-igw'. Dem Gateway geben wir zusätzlich Tags, um später die Suche von diesem zu erleichtern.

**Create internet gateway** [Info](#)  
An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

**Internet gateway settings**  
**Name tag**  
Creates a tag with a key of 'Name' and a value that you specify.

**Tags - optional**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - optional	
<input type="text" value="Q Name"/> <a href="#">×</a>	<input type="text" value="Q semester-igw"/> <a href="#">×</a>	<a href="#">Remove</a>
<input type="text" value="Q Environment"/> <a href="#">×</a>	<input type="text" value="Q Semester"/> <a href="#">×</a>	<a href="#">Remove</a>

[Add new tag](#)  
You can add 48 more tags.

[Cancel](#) [Create internet gateway](#)

Abbildung 5: Internet Gateway erstellen

Das Gateway ist erstellt, aber noch keinem VPC zugewiesen. Beim Gateway unter 'Actions -> Attach to VPC' fügen wir diesen hinzu.

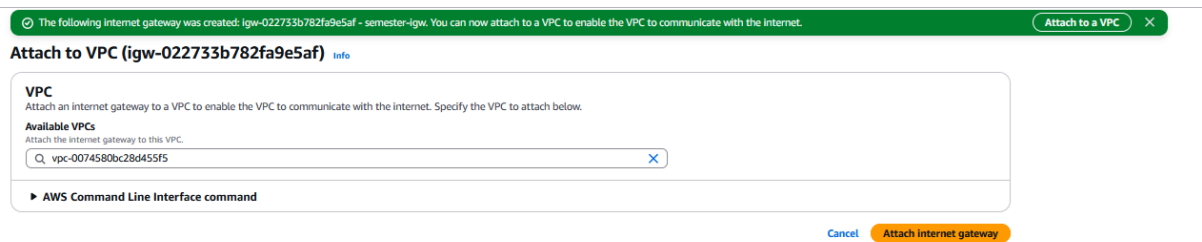


Abbildung 6: Internet Gateway einem VPC hinzufügen

### 9.1.2 Routing Table

Jetzt aktualisieren wir die neue Routing Table, damit das Gateway Internetzugriff hat. In dem Tab 'Route Tables' editieren wir den Table mit dem Namen 'rtb-public-semester'.

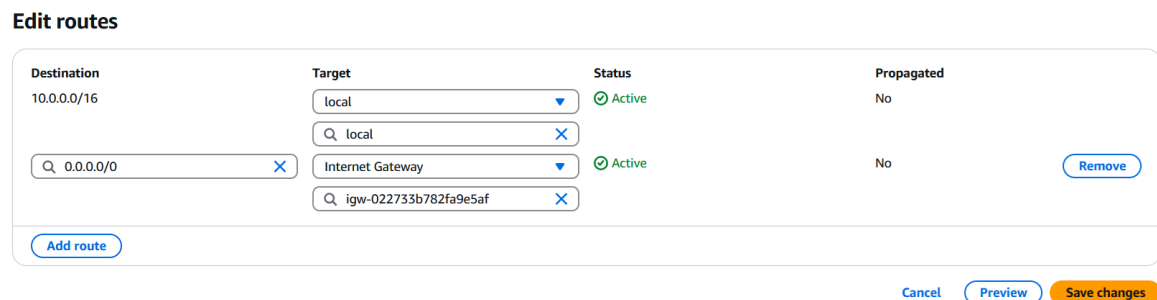


Abbildung 7: Routen des Public Route Tables anpassen

## 9.2 Enabling Internet Access for Private Subnet

Das private Subnet hat keinen direkten Zugriff, sondern nutzt einen NAT Gateway vom Public Subnet.

### 9.2.1 NAT Gateway

**Routing Table** Wie beim Public Subnet, muss man nun die Routing Table aktualisieren, nur für den NAT Gateway anstatt des Internet Gateways.

Create NAT gateway [info](#)

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

NAT gateway settings

Name - optional

Create a tag with a key of 'Name' and a value that you specify.

public-nat

The name can be up to 256 characters long.

Subnet

Select a subnet in which to create the NAT gateway.

subnet-08ad7172adeb52985 (semester-public-sn)

Connectivity type

Select a connectivity type for the NAT gateway.

☒ Public

☐ Private

Elastic IP allocation ID [info](#)

Assign an Elastic IP address to the NAT gateway.

eipalloc-0a25fdd18d517ee2d

Allocate Elastic IP

Additional settings [info](#)

Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Q Name

X

Value - optional

Q public-nat

X

Remove

Q Environment

X

Q Semester

X

Remove

Add new tag

You can add 48 more tags.

Cancel

Create NAT gateway

Abbildung 8: Erstellung des NAT Gateways

nat-0f921e9ea877c68ce / public-nat

Actions

Details

NAT gateway ID

nat-0f921e9ea877c68ce

NAT gateway ARN

arn:aws:ec2:us-east-1:766687047518:natgateway/nat-0f921e9ea877c68ce

VPC

vpc-0074580bc28d455f5 / semester-vpc

Connectivity type

Public

Primary public IPv4 address

-

Subnet

subnet-08ad7172adeb52985 / semester-public-sn

State

Pending

Primary private IPv4 address

10.0.1.195

Created

Thursday, December 19, 2024 at 12:06:35 GMT+1

State message [info](#)

-

Primary network interface ID

eni-0eb3edaa26ea49492

Deleted

-

Secondary IPv4 addresses

Monitoring

Tags

Secondary IPv4 addresses

Search

Edit secondary IPv4 address associations

Private IPv4 address

Network interface ID

Status

Failure message

Secondary IPv4 addresses are not available for this nat gateway.

Abbildung 9: Erstellung des NAT Gateways

Edit routes

Destination

10.0.0/16

Target

local

X

Status

Active

Propagated

No

Q 0.0.0/0

X

NAT Gateway

X

Active

No

Remove

Add route

Cancel

Preview

Save changes

Abbildung 10: Routen des Private Route Tables anpassen

14

## 10 Security Groups

Für das Projekt haben wir mehrere Security Groups erstellt.

### 10.1 Public GitLab

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	80	0.0.0.0/0	HTTP
Inbound	TCP	443	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH(GitLab)
Inbound	TCP	2424	My IP	SSH(Admin)
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 6: Security Group: Public GitLab

### 10.2 Private GitLab

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	80	0.0.0.0/0	HTTP
Inbound	TCP	443	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 7: Security Group: Private GitLab

### 10.3 DNS

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	53	0.0.0.0/0	HTTP
Inbound	TCP	53	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 8: Security Group: DNS

## 10.4 LDAP

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	53	0.0.0.0/0	HTTP
Inbound	TCP	53	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 9: Security Group: LDAP

## 11 Launch Instance

Die folgenden Screenshots zeigen, wie wir zwei Ubuntu-Instanzen für Primary und Secondary DNS erstellen können.



## Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags [Info](#)

Key [Info](#)

Q Name X

Value [Info](#)

Q DNS-Primary X

Resource types [Info](#)

Select resource types ▼

Remove

Instances X

Key [Info](#)

Q Environment X

Value [Info](#)

Q Semester X

Resource types [Info](#)

Select resource types ▼

Remove

Instances X

Add new tag

You can add up to 48 more tags.

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Search

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0e2c8caa4b6378d8c (64-bit (x86)) / ami-0932ffb346ea84d48 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Username

Verified provider

64-bit (x86)

ami-0e2c8caa4b6378d8

ubuntu

Verified provider

Abbildung 11: Instanzerstellung vom Primary DNS

17

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

vockey

Create new key pair

▼ Network settings Info

VPC - required Info

vpc-0074580bc28d455f5 (semester-vpc)

10.0.0.0/16

Create new VPC

Subnet Info

subnet-066defd884846189d semester-private-sn

VPC: vpc-0074580bc28d455f5 Owner: 766687047518 Availability Zone: us-east-1f

Zone type: Availability Zone IP addresses available: 251 CIDR: 10.0.2.0/24

Create new subnet

Auto-assign public IP Info

Disable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Common security groups Info

Select security groups

DNSSG sg-07020661e295f0536

VPC: vpc-0074580bc28d455f5

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

► Advanced network configuration

Abbildung 12: Instanzerstellung vom Primary DNS

18

## Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags [Info](#)

Key [Info](#)

Q Name X

Value [Info](#)

Q DNS-Secondary X

Resource types [Info](#)

Select resource types ▼

Instances X

Remove

Key [Info](#)

Q Environment X

Value [Info](#)

Q Semester X

Resource types [Info](#)

Select resource types ▼

Instances X

Remove

Add new tag

You can add up to 48 more tags.

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUSE

Debian

debian

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-0e2c8caa4b6378d8c (64-bit (x86)) / ami-0932ffb346ea84d48 (64-bit (Arm))  
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).  
Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

AMI ID

Username

64-bit (x86) ▼

ami-0e2c8caa4b6378d8

ubuntu

Verified provider

Abbildung 13: Instanzerstellung vom Secondary DNS

19

▼ Instance type

Info | Get advice

Instance type

t2.micro

Free tier eligible

Family: t2

1 vCPU

1 GiB Memory

Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login)

Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

vockey

Create new key pair

▼ Network settings

Info

VPC - required

Info

vpc-0074580bc28d455f5 (semester-vpc)

10.0.0.0/16

Create new vpc

Subnet

Info

subnet-066defd884846189d

semester-private-sn

VPC: vpc-0074580bc28d455f5

Owner: 766687047518

Availability Zone: us-east-1f

Zone type: Availability Zone

IP addresses available: 250

CIDR: 10.0.2.0/24

Create new subnet

Auto-assign public IP

Info

Disable

Firewall (security groups)

Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Common security groups

Info

Select security groups

DNSSG sg-07020661e295f0536

VPC: vpc-0074580bc28d455f5

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

► Advanced network configuration

Abbildung 14: Instanzerstellung vom Secondary DNS

## 12 SSH Access zu Server im Private Subnet

Weil die Server in dem private Subnet keine public IPV4 Adresse haben, können wir nicht auf sie direkt mit SSH zugreifen. Wir können um das doch zu erreichen einen Umweg über den GitLab Server nehmen, welcher sich im public Subnet befindet und eine public IPV4 Adresse hat.

### 12.1 Setting up SSH Agent Forwarding

Wir müssen zuerst den SSH Key, den wir von AWS Academy bekommen haben hinzufügen.

```
ssh-add. ~/.ssh/labsuser.pem
```

Danach ermöglichen wir SSH Agent Forwarding durch Änderung des files `~/.ssh/config`:

```
Host example.com
ForwardAgent yes
```

Nun müssen wir example.com mit der Public IPV4 Adresse von GitLab ersetzen. Dies muss bei jedem Neustart des GitLab Servers wiederholt werden, weil sich die IP-Adresse bei jedem Start ändert.

## 12.2 SSH Zugriff

Wir können jetzt SSH verwenden um den GitLab Server zu erreichen.

```
ssh ubuntu@93.71.270.50
```

Und auch um vom GitLab Server zu einem Private Subnet Server springen.

```
ssh ubuntu@10.0.2.225(Public IP des DNS-Primary EC2 Servers)
```

## 12.3 Aktualisieren des Betriebssystems

Sobald wir auf dem Primary DNS Server sind müssen wir zuerst sicherstellen, dass das Betriebssystem auf dem neusesten Stand ist. Um das zu erreichen führen wir die folgenden Commands aus.

```
sudo apt update && sudo apt upgrade && sudo apt full-upgrade
sudo reboot
```

# 13 Setup der DNS Server

Nun konfigurieren wir die beiden DNS Server.

## 13.1 Installation von BIND9

Nachdem wir sichergegangen sind, dass das Betriebssystem auf dem neuesten Stand ist installieren wir BIND9. Eine Software Package mit welchem Namens Resolution machen können. Das erreichen wir mit dem folgenden command:

```
sudo apt install bind9 bind9utils bind9-doc
```

Wir setzen nun BIND in den IPV4 Modus. Das erreichen wir indem wir das file */etc/default/named* ändern und -4 am Ende vom OPTIONS Parameter hinzufügen.

```
OPTIONS="--u bind -4"
```

Zuletzt starten wir BIND9 neu.

```
sudo systemctl restart bind9
```

## 13.2 Konfiguration des Primary DNS Servers

Zunächst konfigurieren wir den Primary DNS Server, welcher auf der EC2 Instanz läuft.

### 13.2.1 Erstellung der Access Contol List

Zuerst müssen wir eine ACL (Access Contol List) für den Primary DNS Server erstellen. Dadurch können wir kontrollieren, wer Zugriff auf den Name Server hat. Die ACL kann in *named.conf.options* geändert werden:

```
sudo nano /etc/bind/named.conf.options
```

Über dem Options-Block erstellen wir eine neue Access Control List namens *trusted*. In dieser benennen wir alle Clients denen wir Zugriff auf den Name Server geben wollen.

```
acl "trusted" {  
    10.0.2.184  
    10.0.2.225  
    10.0.1.246  
    10.0.2.10  
};
```

### 13.2.2 Konfiguration der Allgemeine Optionen

Nun editieren wir den Options Block in dem *named.conf.options* File um generelle Einstellungen festzulegen.

```
options {  
    directory "/var/cache/bind";  
    recursion yes; # enables recursive queries  
    allow-recursion { trusted; }; # allows recursive queries from "trusted" clients  
    listen-on { 10.0.0.0/16; }; # VPC CIDR - listen on private network only  
    allow-transfer { none; }; # disable zone transfers by default  
    forwarders {  
        8.8.8.8;  
        8.8.4.4;  
    };  
    ...  
};
```

Wenn der DNS Server eine Anfrage bekommt, welche er nicht selbst beantworten kann, schickt er diese an den unter *forwarders* definierten Name Server.

### 13.2.3 Konfiguration des “Local” Files

In der Local File geben wir die Forward- und Reverse-Zonen an, die den Bereich für die Verwaltung und Definition von DNS-Einträgen festlegen.

```
sudo nano /etc/bind/named.conf.local
```

### 13.2.4 Hinzufügen der Forward Zone

```
zone "semesterDevOps.com" {  
    type primary;  
    file "/etc/bind/zones/db.semesterDevOps.com";  
    allow-transfer { 10.0.2.225; };  
};
```

*type primary* → definiert diesen Server als den Primary Name Server für die Zone.

*File* → Pfad zum Zone File

*allow-transfer* → IP Adressen des sekundären Servers welchem es erlaubt ist, von diesem Server zu transferieren.

### 13.2.5 Hinzufügen der Reverse Zone für das Public Subnet

```
zone "1.0.10.in-addr.arpa" {  
    type primary;  
    file "/etc/bind/zones/db.10.0.1"; # 10.0.1.0/24 Subnet  
    allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary  
};
```

Um Reverse Lookups zu erlauben, müssen wir die Reverse Zone so konfigurieren, dass wir in der Range unseres Public Subnets sind.

### 13.2.6 Hinzufügen der Reverse Zone für das Private Subnet

```
zone "2.0.10.in-addr.arpa" {  
    type primary;  
    file "/etc/bind/zones/db.10.0.2"; # 10.0.2.0/24 Subnet  
    allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary  
};
```

Dasselbe machen wir für das Private Subnet.

### 13.2.7 Erstellen des Forward Zone Files

Die Forward-Zone Datei ist der Ort, an dem wir DNS-Einträge für Forward-DNS-Abfragen definieren. Das bedeutet, wenn der DNS eine Namesabfrage erhält, wird er in der Forward-Zonen Datei nachschauen, um die entsprechende private IP-Adresse von host1 zu ermitteln. Zuerst müssen wir das Directory erstellen. Hier werden wir auch die Forward Zone Files einfügen:

```
sudo mkdir /etc/bind/zones
```

Jetzt erstellen wir das Zone File:

```
sudo nano /etc/bind/zones/db.semesterDevOps.com  
  
;  
; BIND data file for local loopback interface  
;  
$TTL 604800  
@ IN SOA ns1.semesterDevOps.com. admin.semesterDevOps.com. (  
    3 ; Serial  
    604800 ; Refresh  
    86400 ; Retry  
    2419200 ; Expire  
    604800 ) ; Negative Cache TTL
```

```

;

; name servers - NS records
IN NS ns1.semesterDevOps.com.
IN NS ns2.semesterDevOps.com.

; name servers - A records
ns1.semesterDevOps.com. IN A 10.0.2.225
ns2.semesterDevOps.com. IN A 10.0.2.10

; 10.0.0.0/16 - A records
gitlab.semesterDevOps.com. IN A 10.0.1.246
glrunner.semesterDevOps.com. IN A 10.0.2.184

```

### 13.2.8 Erstellen des Reverse Zone Files für das Public Subnet

Reverse-Zonen Dateien sind der Ort, an dem wir DNS PTR-Einträge für Reverse-DNS-Abfragen definieren. Das bedeutet, wenn der DNS eine Abfrage nach einer IP-Adresse erhält, z. B. 10.0.1.77, wird er in der Reverse-Zonen Datei nachschauen, um den entsprechenden FQDN zu ermitteln, z.B. gitlab.semesterDevOps.com in diesem Fall. Lassen Sie uns die Zonendatei erstellen:

```

sudo nano /etc/bind/zones/db.10.0.1

;
; BIND reverse data file for 10.0.1.0/24
;
$TTL 604800
@ IN SOA ns1.semesterDevOps.com. admin.semesterDevOps.com. (
    1 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800
) ; Negative Cache TTL
;
; Name servers - NS records
IN NS ns1.semesterDevOps.com.
IN NS ns2.semesterDevOps.com.
; PTR Records
77 IN PTR gitlab.semesterDevOps.com. ; 10.0.1.246

```

- semesterDevOps.com. oder @ : Wurzel der Zone. Dies gibt an, dass die Zonendatei für die Domain semesterDevOps.com bestimmt ist. @ ist nur ein Platzhalter, der den Inhalt der \$ORIGIN-Variable ersetzt.
- IN SOA: Der Teil „IN“ steht für Internet. SOA ist der Hinweis darauf, dass dies ein Start of Authority-Eintrag ist.
- ns1.semesterDevOps.com.: Definiert den primären Name Server für diese Domain.



- `admin.semesterDevOps.com.`: E-Mail-Adresse des Administrators für diese Zone. Das „@“ wird in der E-Mail-Adresse durch einen Punkt ersetzt.
- `Serial`: Seriennummer für die Zone Datei. Jedes Mal, wenn die Zone Datei bearbeitet wird, muss diese Nummer inkrementiert werden, damit die Zonendatei korrekt propagiert wird. Sekundäre Server prüfen, ob die Seriennummer der Zone auf dem primären Server größer ist als die, die sie auf ihrem System haben. Ist dies der Fall, fordert der sekundäre Server die neue Zonendatei an; wenn nicht, wird weiterhin die ursprüngliche Datei bereitgestellt.
- `Refresh`: Aktualisierungsintervall für die Zone. Dies ist der Zeitraum, den der sekundäre Server wartet, bevor er den primären Server auf Änderungen der Zonendatei abfragt.
- `Retry`: Wiederholungsintervall für diese Zone. Wenn der sekundäre Server nach Ablauf des Aktualisierungsintervalls keine Verbindung zum primären Server herstellen kann, wartet er diesen Zeitraum und versucht dann erneut, den primären Server abzufragen.
- `Expire`: Ablaufzeitraum. Wenn ein sekundärer Name Server den primären Server für diesen Zeitraum nicht kontaktieren konnte, gibt er keine Antworten mehr als autoritative Quelle für diese Zone zurück.
- `Negative Cache TTL`: Zeitraum, für den der Name Server einen Namensfehler zwischenspeichert, wenn der angeforderte Name in dieser Datei nicht gefunden werden kann.

### 13.2.9 Erstellen des Reverse Zone Files für das Private Subnet

Nun das selbe für das Private Subnet.

```
sudo nano /etc/bind/zones/db.10.0.2
```

```
;
; BIND reverse data file for 10.0.2.0/24
;
$TTL 604800
@ IN SOA semesterDevOps.com. admin.semesterDevOps.com. (
    1 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800
) ; Negative Cache TTL
;
; Name servers - NS records
IN NS ns1.semesterDevOps.com.
IN NS ns2.semesterDevOps.com.
; PTR Records
252 IN PTR ns1.semesterDevOps.com. ; 10.0.2.225
217 IN PTR ns2.semesterDevOps.com. ; 10.0.2.10
148 IN PTR glrunner.semesterDevOps.com. ; 10.0.2.184
```

**Überprüfen der BIND Configuration Syntax** Wir können die Syntax aller *named.conf* Files mit dem folgenden Command überprüfen:

```
sudo named-checkconf
```

Um die Zone Files zu überprüfen helfen uns diese Commands:

```
sudo named-checkzone 1.0.10.in-addr.arpa /etc/bind/zones/db.10.0.1
sudo named-checkzone 2.0.10.in-addr.arpa /etc/bind/zones/db.10.0.2
```

Sobald die Syntax überprüft wurde, müssen wir BIND9 neustarten, damit die Änderungen übernommen werden:

```
sudo systemctl restart bind9
```

## 14 Konfiguration des sekundären DNS-Servers

Der sekundäre DNS-Server wird alle Einträge des primären DNS-Servers spiegeln und alle Anfragen beantworten, wenn der primäre DNS-Server nicht verfügbar ist.

## 14.1 Einrichten der Zugriffskontrollliste und allgemeiner Optionen

Sowie der erste DNS, braucht der Zweite ebenfalls eine Access Control List.

```
acl "trusted" {
    10.0.2.225
    10.0.2.10
    10.0.1.246
    10.0.2.184
};

options {
    directory "/var/cache/bind";
    recursion yes; # enables recursive queries
    allow-recursion { trusted; }; # allows recursive queries from "trusted" clients
    listen-on { 10.0.0.0/16; }; # VPC CIDR - listen on private network only
    allow-transfer { none; }; # disable zone transfers by default
    forwarders {
        8.8.8.8;
        8.8.4.4;
    };
    ...
};
```

## 14.2 Konfigurieren vom Local File

Die Local Files vom zweiten DNS wird genau so konfiguriert wie vom Ersten. Es gibt nur kleine Unterschiede wie `type secondary` statt `type primary`.

```
zone "semesterDevOps.com" {
    type secondary;
    file "/etc/bind/zones/db.semesterDevOps.com";
    allow-transfer { 10.0.2.225; };
};

zone "1.0.10.in-addr.arpa" {
    type secondary;
    file "/etc/bind/zones/db.10.0.1"; # 10.0.1.0/24 Subnet
    allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary
};

zone "2.0.10.in-addr.arpa" {
    type secondary;
    file "/etc/bind/zones/db.10.0.2"; # 10.0.2.0/24 Subnet
    allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary
};
```

Zum Überprüfen der Configuration verwenden wir:

```
sudo named-checkconf
```

Wenn alles funktioniert, dann starten wir BIND9 erneut:

```
sudo systemctl restart bind9
```

## 15 Konfiguration der Server

Unten dargestellt ist die Konfiguration des DNS-Servers auf dem Public-GitLab-Server. Es ist jedoch wichtig zu beachten, dass diese Konfiguration auf allen Servern in der Infrastruktur umgesetzt werden muss. Zuerst müssen wir herausfinden, welches Gerät mit unserem privaten Netzwerk verbunden ist:

```
ip address show to 10.0.0.0/16
```

### 15.1 Festlegen der Nameserver

In Ubuntu werden alle DNS-Dienste über **systemd** verwaltet. Der für DNS verantwortliche Dienst heißt **systemd-resolved** und enthält das Dienstprogramm **resolvectl**, mit dem einige DNS-Konfigurationen vorgenommen werden können.

Um die Nameserver für DNS-Abfragen auf unsere eigenen DNS-Nameserver festzulegen, können wir den folgenden Befehl verwenden:

```
resolvectl dns <Interface> 10.0.2.225 10.0.2.10
```

Hierbei steht **<Interface>** für die Netzwerkschnittstelle, zum Beispiel **eth0**. Die IP-Adresse **10.0.2.225** gehört zum primären DNS-Server, und **10.0.2.10** ist die IP-Adresse des sekundären DNS-Servers.

### 15.2 Änderungen dauerhaft speichern

Jedoch speichert dieser Command die Änderungen der Einstellungen nicht persistent. Sie würden zum Beispiel nach einem Reboot verloren gehen. Um die Änderungen permanent zu machen müssen wir ein paar config-Files ändern:

```
sudo nano /etc/systemd/resolved.conf
```

Spezifisch das File *etc/systemd/resolved.conf* möchten wir ändern. Zunächst müssen wir die folgende Zeile anpassen:

```
[Resolve]
DNS=10.0.2.225 10.0.2.10
```

Wieder müssen hier IP Adressen des Primary und Secondary DNS angegeben werden. Damit die Änderungen eintreten müssen wir den *systemd-resolved service* restarten:

```
sudo systemctl restart systemd-resolved
```

Gehe sicher, dass */etc/resolv.conf* mit dem Stub-Resolver von *systemd-resolved* verknüpft ist:

```
sudo ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

Mit dem Folgenden Command kann nun auch überprüft werden, ob das System nun wirklich unseren DNS Server verwendet:

```
sudo resolvectl status
```

### 15.3 Testen der Nameserver

Wir können überprüfen, ob unsere Nameserver funktionieren, indem wir einige Abfragen durchführen. Wir beginnen mit DNS-Forward-Lookups:

```
nslookup gitlab.semesterDevOps.com
```

Sobald wir bestätigt haben, dass diese ordnungsgemäß funktionieren, führen wir auch DNS-Reverse-Lookups durch:

```
nslookup glrunner.semesterDevOps.com
```

Alternativ können wir auch den Befehl dig verwenden, um detailliertere Ausgaben zu erhalten:

```
nslookup 10.0.1.246
```

```
nslookup 10.0.2.184
```

### 15.4 Überprüfen, ob der sekundäre DNS-Server wie vorgesehen funktioniert

Es gibt mehrere Möglichkeiten, um zu testen, ob der sekundäre DNS-Server wie vorgesehen arbeitet:

#### 1. Direkte Abfrage des sekundären DNS-Servers mit dig

```
dig @10.0.2.10 gitlab.semesterDevOps.com
```

#### 2. DNS-Primary-Instanz stoppen und Abfragen testen

Stoppen Sie die EC2-Instanz DNS-Primary und überprüfen Sie, ob DNS-Abfragen mithilfe von nslookup oder dig beantwortet werden können.

#### 3. Lokalen DNS-Cache auf dem Client deaktivieren

Bearbeiten Sie dazu die Datei /etc/systemd/resolved.conf und fügen Sie die folgende Zeile hinzu:

```
Cache=no
```

## 16 GitLab

### 16.1 Konfiguration des SSH-Ports

GitLab erlaubt es Benutzern, über SSH auf Repositories zuzugreifen. Dies führt jedoch zu Konflikten mit dem standardmäßigen SSH-Zugang des Systems. Daher ändern wir den SSH-Port des Systems auf 2424, sodass der Port 22 für GitLab verwendet werden kann. Diese Konfiguration wurde gewählt, da das Klonen eines Repositories eine häufigere Aufgabe ist als der SSH-Zugriff auf den Server. Durch die Verwendung des Standard-Ports für das Repository-Klonen entfällt die Notwendigkeit, einen benutzerdefinierten Port anzugeben. Für den selteneren SSH-Zugriff muss der alternative Port explizit angegeben werden. Die Änderung wird in der Datei `/etc/ssh/sshd_config` vorgenommen:

```
Port = 2424
```

### 16.2 Anpassen der Sicherheitsgruppe auf AWS

Damit eine Verbindung über den Port 2424 möglich ist, muss dieser Port in der Sicherheitsgruppe `PublicGitLabSecurityGroup` freigegeben werden. Beachten Sie, dass bei jedem SSH-Zugriff auf die GitLab-Instanz der alternative Port angegeben werden muss:

```
ssh ubuntu@98.80.148.9 -p 2424
```

### 16.3 Installation von GitLab CE mit Docker Compose

Im Anschluss wird Docker gemäß der offiziellen Dokumentation installiert. Nach erfolgreichem Test mit dem `hello-world`-Image folgen wir der GitLab-Dokumentation für die Installation.

#### 16.3.1 Erstellen von Verzeichnissen zur Datenpersistenz

Gemäß der Dokumentation erstellen wir ein Verzeichnis für Konfigurations-, Log- und Daten-Dateien:

```
sudo mkdir -p /srv/gitlab
sudo chown -R ubuntu:ubuntu /srv/gitlab
export GITLAB_HOME=/srv/gitlab
```

### 16.3.2 Erstellen des Containers mit Docker Compose

Wir verwenden Docker Compose, um GitLab zu installieren. Dazu erstellen wir eine Datei `docker-compose.yml` mit folgendem Inhalt:

```
services:
  gitlab:
    image: gitlab/gitlab-ce:latest
    container_name: gitlab
    restart: always
    hostname: 'gitlab.semesterDevOps.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://98.80.148.9'
    ports:
      - '80:80'
      - '443:443'
      - '22:22'
    volumes:
      - '$GITLAB_HOME/config:/etc/gitlab'
      - '$GITLAB_HOME/logs:/var/log/gitlab'
      - '$GITLAB_HOME/data:/var/opt/gitlab'
    shm_size: '256m'
```

Die `external_url` muss mit der öffentlichen IPv4-Adresse der GitLab-EC2-Instanz aktualisiert werden. Da die IP-Adresse nach jedem Neustart wechselt, ist eine Anpassung der Datei `docker-compose.yml` nach jedem Neustart notwendig. Zum Starten des Containers führen wir folgenden Befehl aus:

```
docker compose up -d
```

GitLab ist nach kurzer Zeit unter `http://98.80.148.9` erreichbar. Der Standard-Benutzername lautet `root`, das Passwort kann mit folgendem Befehl abgerufen werden:

```
docker compose exec -it gitlab cat /etc/gitlab/initial_root_password
```

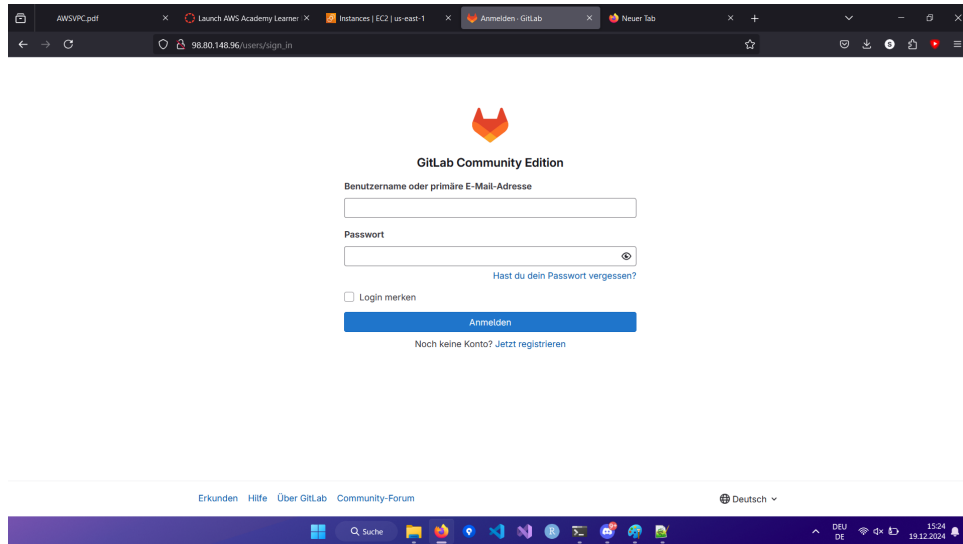


Abbildung 15: GitLab läuft



## 17 GitLab Runner

Als nächstes installieren wir GitLab Runner auf der GitLab-Runner-EC2-Instanz. Wie zuvor verbinden wir uns per SSH mit der Maschine, aktualisieren alle Systempakete, starten die Maschine neu und installieren anschließend Docker gemäß der offiziellen Dokumentation.

```
docker pull gitlab/gitlab-runner:latest
```

### 17.1 Installation des GitLab Runners mit Docker

Zuerst laden wir das neueste Docker-Image herunter. Anschließend erstellen wir ein neues Docker-Volume, um die Konfiguration auch bei Neustarts des Containers zu behalten.

```
docker volume create gitlab-runner-config
```

Danach starten wir den GitLab-Runner-Container:

```
docker run -d --name gitlab-runner --restart always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v gitlab-runner-config:/etc/gitlab-runner \
gitlab/gitlab-runner:latest
```

### 17.2 Registrierung des Runners in GitLab

Um den Runner zu registrieren, folgen wir der entsprechenden Dokumentation. GitLab-Runner können entweder Projekten, Gruppen oder der gesamten Instanz zugeordnet werden. Wir werden unseren Runner mit der gesamten Instanz verknüpfen.

#### 17.2.1 Erstellen des Runner-Authentifizierungstokens

Um ein Authentifizierungstoken für den Runner zu erhalten, müssen wir über die GitLab-Weboberfläche einen Instanz-Runner hinzufügen. Wie das genau funktioniert, ist in der offiziellen Dokumentation beschrieben.

#### 17.2.2 Registrierung des Runners

Um einen neuen Runner zu registrieren, führen wir folgenden Befehl (auf der GitLab-Runner-EC2-Instanz) aus:

```
docker run --rm -it -v gitlab-runner-config:/etc/gitlab-runner gitlab/gitlab-runner
register
```

Nach Ausführung dieses Befehls werden wir aufgefordert, einige Einstellungen vorzunehmen. Diese konfigurieren wir wie folgt:

- GitLab-Instanz-URL:  
Hier geben wir `http://gitlab.SemesterDevOps.com` ein. Diese URL wird von unserem DNS-Server auf den Server aufgelöst, auf dem die GitLab-Instanz läuft.

- Runner-Authentifizierungstoken:  
Hier geben wir den Token ein, das in der GitLab-Weboberfläche angezeigt wird.
- Executor:  
Wir wählen `docker`.
- Standard-Docker-Image:  
Wir geben `alpine:latest` ein.

## Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. [Not sure which one to select?](#)

## Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a [system](#) or [user service](#).

 **You've registered a new runner!**

Your runner is online and ready to run jobs.

To view the runner, go to [Admin area](#) > [Runners](#).

[View runners](#)

Abbildung 16: GitLab Runner registriert

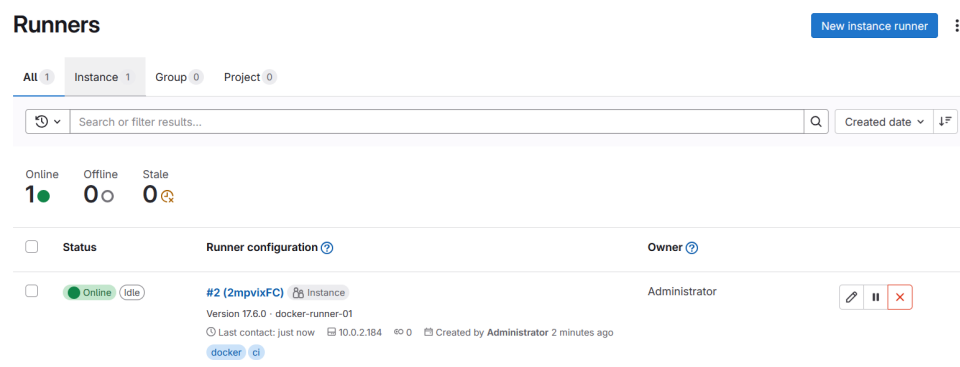


Abbildung 17: GitLab Runner ist Online

# 18 LDAP Server

## 18.1 Vorbereitung

Um den LDAP-Server zu installieren, müssen wir zuerst eine neue EC2-Instanz erstellen und die Sicherheitsgruppe LDAP erstellen.

### 18.1.1 Instance erstellen

- **Name:** LDAP
- **OS:** Latest LTS Release of Ubuntu → Ubuntu 24.04
- **Instance Type:** t2.micro
- **VPC:** vpc-devops
- **Subnet:** sn-1a-private-devops (Private Subnet)
- **Public IP:** No
- **Security Group:** LDAPSecurityGroup

### 18.1.2 Security Group

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	389	0.0.0.0/0	LDAP
Inbound	TCP	639	0.0.0.0/0	LDAP over SSL
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 10: Security Group: LDAP

### 18.1.3 DNS Eintrag für LDAP Server

In den DNS Server Primary und Secondary muss in Zugriffskontrollliste die IP Adresse des LDAP Servers hinzugefügt werden. Siehe oben im Abschnitt 14.1.

### 18.1.4 Zone Files

Im Primary DNS Server müssen die Zone Files für die LDAP Domain hinzugefügt werden. Siehe oben im Abschnitt 13.2.4 und folgende.

## 18.2 LDAP

Als nächstes richten wir den OpenLDAP-Server ein. Dazu verbinden wir uns über SSH mit der LDAP-EC2-Instanz, aktualisieren alle Systempakete und starten das System neu. Dann installieren wir den LDAP-Server und die Kommandozeilen-Utilities:

```
sudo apt install slapd ldap-utils
```

Während dieses Prozesses werden wir aufgefordert, ein Passwort für den LDAP-Admin-Benutzer festzulegen. Wir entscheiden uns für "admin", damit es einfacher zu merken ist.

Als nächstes ändern wir den Directory Information Tree (DIT), um ihn an unseren Domainnamen anzupassen:

```
sudo dpkg-reconfigure slapd
```

Um das DIT-Suffix auf `dc=semesterDevOps,dc=com` zu ändern, geben wir `ilovedevops.com` ein, wenn wir nach dem DNS-Domainnamen gefragt werden.

Als nächstes konfigurieren wir einige Standardwerte, um uns später Tipparbeit zu ersparen. Führen Sie aus:

```
sudo nano /etc/ldap/ldap.conf
```

und fügen Sie diese Zeilen hinzu:

### 18.2.1 Verzeichnis füllen

Lassen Sie uns eine Organisationseinheit für 'People' erstellen, in der wir eine Person namens Samuel hinzufügen. Um die Einträge in das Verzeichnis hinzuzufügen, verwenden wir das LDAP Data Interchange Format (LDIF). Dies ist ein standardisiertes Textformat, das verwendet wird, um Daten für LDAP-Verzeichnisse zu definieren. Erstellen Sie eine neue LDIF-Datei und nennen Sie sie beispielsweise

```
add_content.ldif
```

Der genaue Dateiname ist nicht entscheidend, solange er später korrekt referenziert wird. Mit dem Kommando `sudo nano add_content.ldif` wird die LDIF-Datei für eine Person geändert:

```
dn: ou=People,dc=semesterDevOps,dc=com
objectClass: organizationalUnit
ou: People
dn: uid=Samuel,ou=People,dc=semesterDevOps,dc=com
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: Samuel
sn: Hammerschmidt
givenName: Samuel
cn: Samuel Hammerschmidt
displayName: Samuel Hammerschmidt
uidNumber: 10000
gidNumber: 5000
userPassword: {CRYPT}x
gecos: Samuel Hammerschmidt
loginShell: /bin/bash
homeDirectory: /home/samuel
```

Wir können die in dieser Datei angegebenen Informationen dem LDAP-Verzeichnis hinzufügen, indem wir folgenden Befehl ausführen:

```
ldapadd -x -D cn=admin,dc=semesterDevOps,dc=com -W -f add_content.ldif
```

Wir können überprüfen, ob alle Daten erfolgreich hinzugefügt wurden, indem wir folgenden Befehl ausführen:

```
ldapsearch -x -LLL -b dc=semesterDevOps,dc=com '(uid=Samuel)' cn gidNumber
```

Nun müssen wir ein gültiges Passwort für den Benutzer Samuel festlegen. Wir haben bisher nur CRYPTx als Platzhalter verwendet, um ein Konto ohne Passwort zu erstellen. (Streng genommen ist CRYPTx auch ein gültiges Passwort. Allerdings gibt es keine Eingabe für die Hash-Funktion, die zu der Ausgabe x führt, was bedeutet, dass wir im Grunde ein ungültiges Passwort definiert haben.) Um das Passwort in ein gültiges zu ändern, können wir ldappasswd verwenden:

```
ldappasswd -x -D cn=admin,dc=semesterDevOps,dc=com -W -S
uid://USERNAME,ou=people,dc=semesterDevOps,dc=com
Passwort: //PASSWORD
```

### 18.2.2 LDAP-Server Verbindung testen

Schließlich können wir testen, ob der LDAP-Server von anderen Instanzen aus erreichbar ist. Wechseln wir zur GitLab-Instanz, installieren LDAP-Utills und führen aus:

```
ldapsearch -x -LLL -H ldap://ldap.semesterDevOps.com -b dc=semesterDevOps,dc=com
'(uid:// USERNAME)' cn gidNumber
```

Wir können weitere Benutzer hinzufügen, indem wir die erforderlichen Schritte beliebig oft wiederholen.

## 18.3 Integration von LDAP mit GitLab

Als nächstes werden wir LDAP mit GitLab integrieren. Dazu folgen wir der offiziellen Dokumentation. Zunächst müssen wir einige Einstellungen konfigurieren, die wir über Umgebungsvariablen in Docker vornehmen. Bearbeiten wir also erneut die docker-compose.yml unseres LDAP-Servers:

```
services:
  gitlab:
    image: gitlab/gitlab-ce:latest
    container_name: gitlab
    restart: always
    hostname: 'gitlab.semesterDevOps.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://54.89.164.241'
        gitlab_rails['ldap_enabled'] = true
        gitlab_rails['ldap_servers'] = {
          'main' => {
            'label' => 'LDAP',
            'host' => 'ldap.semesterDevOps.com',
            'port' => 389,
            'uid' => 'uid',
            'encryption' => 'plain',
            'verify_certificates' => false,
```

```

        'active_directory' => false,
        'base' => 'dc=semesterDevOps,dc=com',
        'lowercase_usernames' => 'false'
    }
}
ports:
  - '80:80'
  - '443:443'
  - '22:22'
volumes:
  - '$GITLAB_HOME/config:/etc/gitlab'
  - '$GITLAB_HOME/logs:/var/log/gitlab'
  - '$GITLAB_HOME/data:/var/opt/gitlab'
shm_size: '256m'

```

Die genaue Wirkung all dieser Einstellungen ist in der oben verlinkten Dokumentation gut beschrieben. Benutzer können sich nun mit ihrer LDAP-UID und ihrem Passwort anmelden.

# Abbildungsverzeichnis

1	Netzwerktopologie der Infrastruktur . . . . .	4
2	Auswahl der Region . . . . .	10
3	Erstellung des Public Subnet . . . . .	11
4	Auto-Assign aktivieren . . . . .	12
5	Internet Gateway erstellen . . . . .	12
6	Internet Gateway einem VPC hinzufügen . . . . .	13
7	Routen des Public Route Tables anpassen . . . . .	13
8	Erstellung des NAT Gateways . . . . .	14
9	Erstellung des NAT Gateways . . . . .	14
10	Routen des Private Route Tables anpassen . . . . .	14
11	Instanzerstellung vom Primary DNS . . . . .	17
12	Instanzerstellung vom Primary DNS . . . . .	18
13	Instanzerstellung vom Secondary DNS . . . . .	19
14	Instanzerstellung vom Secondary DNS . . . . .	20
15	GitLab läuft . . . . .	32
16	GitLab Runner registriert . . . . .	34
17	GitLab Runner ist Online . . . . .	34

# Tabellenverzeichnis

1	Netzwerkdienste und IP-Zuordnung . . . . .	4
2	Eingehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste . .	5
3	Ausgehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste .	5
4	Server-Spezifikationen: Betriebssystem, Pakete und Instanztypen . . . . .	6
5	Team-Aufgaben und Zuständigkeiten . . . . .	10
6	Security Group: Public GitLab . . . . .	15
7	Security Group: Private GitLab . . . . .	15
8	Security Group: DNS . . . . .	15
9	Security Group: LDAP . . . . .	16
10	Security Group: LDAP . . . . .	35